Cascade Style Sheet

スタイルシート Webデザイン

CSS2完全解説

すみけんたろう著

技術評論社

Cascade Style Sheet

スタイルシートWebデザイン

CSS2完全解説

すみけんたろう著

技術評論社

- 本書掲載のリスト使用によって生じたいかなる損害についても，弊社および著者は一切の責任を負いかねませんので，あらかじめご了承ください．
- 本書に関する質問は，すべて封書でお願いいたします．お電話によるご質問には一切お答えできません．
- 本書中に記載されている会社名，製品名はそれぞれの会社の商標，登録商標，商品名です．
- 本書は表記のバージョンについての解説書で，発行時点の最新バージョンに基づいています．しかし書籍の性質上，時の経過とともにソフトウェアや規格の次のバージョンが出て，いつかは古いバージョンの解説書になるときがきます．本書を購入されるお客様は，ご使用のソフトなどのバージョンを確認してください．
- 掲載のインターネットの接続先情報なども，執筆時点で確認したものですので，本書の記述とは異なっていることもありえますのでご了承ください．

はじめに

筆者がはじめて触れたWWW関連のモノは，MOSAICでした．その後の衝撃的なNetscapeの登場，WWWブラウザの能力の拡張，Windows 95の普及に伴うインターネット利用者の急増などが，ごく短い間に起こりました．

96年12月，W3CはCSS（Cascading Style Sheet）を提唱・推奨しました．97年には，Netscape Navigator4.0やInternet Explorer3.02，4.0がCSSをサポートしはじめました．そして今年の5月12日になってCSS2が勧告されました．未だにCSS2を完全にサポートしたWWWブラウザはありませんが，それも時間の問題だと考えられます．

ようやく我々は，文書を簡潔に記述する方法（HTML）と，その文書のスタイルを調整する方法（CSS）を手に入れました．この2つを組み合わせれば，ワードプロセッサやDTPソフトに対抗できるほどの文書表現が可能です．

本書は，CSSとHTMLを用いて読みやすい文書を作成するための手引きを目指して作られました．したがって，CSSやHTMLをトリッキーに用い派手な効果を得るのではなく，「見出しは見出しらしく，本文は本文らしく」を目標に置いたスタイル指定の考え方を紹介しました．

❶HTML文書に関する知識の整理

HTML文書からスタイル情報を剥奪し，スタイルシートに押し込みます．これにより，HTML文書はより単純で明快になります．

❷スタイルシートを記述する際の考え方の紹介

ただプロパティをいじっただけでは，望ましい使い方はわかりません．理想図をもとにした設計から実際にシートを記述するまでを，例を用いてガイドします．

本書を読むにあたって，HTML文書作成経験やプログラミング記述経験があるほうが望ましいのですが，なくても理解することは可能です．「WWWできれいな文書を公開したい」という意志があれば，例示したスタイルシートを参考に試行錯誤するうちに，スタイルシートをマスターできるでしょう．

本書が完成するまでには，多くの人のご助力をいただきました．

矢野啓介様には初期段階で内容の検討を手伝っていただきました．石野恵一郎様と内田明様は，まとめの段階で例文内のいくつかの文法的な誤りや表現の曖昧さを指摘してくださいました．幾重にもお礼申し上げます．そして，少々偏屈な内容にもかかわらず出版を許可してくださった技術評論社の加藤博編集局長，金田富士男氏に感謝します．

1998年7月吉日

すみけんたろう

Contents

1 Introduction to CSS with HTML 11

2 tutorial of HTML as SGML 29

Some more HTML 53

4 CSS syntax in detail 79

本書で使われている用語の解説と注意事項

用語の解説

◆W3C（WWW Consortium）とは

WWW技術の標準化と推進を目的とする国際学術研究開発組織で，94年に設立されました．マサチューセッツ工科大学計算機科学研究所(MIT/LCS)・フランス国立情報処理自動化研究所(INRIA)・慶應義塾大学SFC研究所（Keio-SFC）の3機関がホストとして共同運営にあたっています．

HTMLやCSSの仕様書は，W3CのWWWサイト（http://www.w3c.org/）から無料で入手できます．本書の内容は，W3Cの公開する仕様書に基づいて書かれています．

◆URL（Uniform Resource Locator）とは

インターネット上のデータなどの位置を表す文字列．http://www.gihyo.co.jp/，ftp://SunSITE.sut.ac.jp/，mailto:foo@bar.comなど．一般に，ファイル名の大文字小文字は区別されるので注意が必要です．URLはRFC1738で定義されています．

◆foo，bar，hoeeeとは

無意味な英単語の例です．架空のe-mailアドレスやファイル名を記述するときに用いることがあります．

◆RFC（Request for Comments）とは

IETF（Internet Engineering Task Force）によって公開された一連の文書で，幅広い話題を取り扱っています．主な話題はInternetやTCP/IPなどの標準化に関するものです．名称こそ「コメントください」ですが，事実上Internetにおけるさまざまな規約の標準を紹介するものと考えられています．たとえば，HTML2.0はRFC1866によって仕様が公開されています．

原文はInterNIC（http://www.internic.net/）が管理しており，無料で入手できます．RFC文書は各所にミラー（複製保管）されており，たとえばRingServerからも入手できます（http://ring.aist.go.jp/，http://ring.nacsis.ac.jp/）．

◆ソースコード，ソースとは

プログラミングの世界では，プログラム言語で記述したプログラム文章そのものをソースコード（源の暗号）とよび，実行できる状態のプログラムと区別します．本書もこれにならい，HTML文書やスタイルシートそのものを表示結果と区別したい場合に，ソースコードという用語を用いることがあります．

注意

◆バージョン番号を付けずにHTML（HyperText Markup Language）と書いてある場合

本書では，HTML仕様にしたがって書かれた文書のことは「HTML文書」と呼称します．HTMLと書いた場合には，記述の仕様・仕組みを意味しています．

W3Cが公開するHTMLの仕様には複数のバージョンがあり，98年7月現在はバージョン4.0が最新です．

本書でHTMLと記述した場合は，バージョンによらない一般性質（たとえば“属性は開始タグ内でのみ指定する”）に言及しています．もし特定のバージョンに依存する性質に言及する場合は，そのバージョン名を明示しています（たとえば“IMG要素のALT属性はHTML4.0では省略できません”）．

◆レベル番号を付けずにCSS（Cascading Style Sheet）と書いてある場合

本書では，構築されたスタイルシートは「スタイルシート」と呼称します．CSSと記述した場合には，CSSの仕様・仕組みを意味しています．

W3Cが公開するCSSの仕様には複数のレベルがあり，今年5月12日のレベル2の勧告によって，CSS1とCSS2とがあります．

本書でCSSと記述した場合は，レベルによらない一般性質（たとえば“宣言の書き方”）に言及しています．もし特定のレベルに依存する性質に言及する場合は，そのレベル名を明示しています（たとえば“このプロパティはCSS1では採用されていませんが，CSS2で採用されています”）．

Introduction to CSS with HTML

1章では，スタイルシートの利点や，HTMLとの役割分担について説明します．同時に，スタイルシートの不在がもたらした混乱についても解説します．

スタイルシートの仕事

1.1.1. 見栄えをコントロールしたい!

WWWの標準ファイルであるHTML文書を作成したことのある方ならば，誰もが一度はこんな不満を抱いたことがあると思います.

「どうしてHTML文書の見栄えは自由に調整**できない**のだろう?」

個人のホームページといっても，大勢の人に公開される作品ですから，その見栄えをきちんとコントロールしたいと考えるのはごく当然です．そしてその不満は，本書で紹介する『スタイルシート』を用いることで**解消**されます.

本章では，詳しい解説は後回しにして，まずはスタイルシートの機能を大まかに紹介しましょう.

1.1.2. スタイルシートでできること

HTML用のスタイルシートとしてW3Cが推奨しているのが「カスケーディング・スタイルシート（CSS)」です．レベル1のCSS（CSS1）だけでも，文書中の任意の部分に関して，以下のプロパティ（property）ごとにスタイルを調整できます.

▶ **フォント系**

フォントファミリー，サイズ，太さ，斜体，スモールキャピタル，行幅

▶ **色と背景効果系**

前景色（文字色)，背景色，背景画像の調整

▶ **テキスト属性系**

単語間隔，文字間隔，下線など，垂直位置（上付き下付き)，水平位置揃え（右揃え，左揃え)，大文字小文字，行頭インデント

▶ **ボックス系**

マージン・パディング，ボーダー（枠線)

▶ **表示位置調整**

表示の横幅，縦幅，フロートと回り込み，回り込みの解除

▶ **その他の表示調整系**

表示形式，空白の取り扱い，リストマークの調整

CSS2では，これらに加えて，「表示位置の絶対的な指定」「自動生成文字（引用符や通し番号）」「印刷への対処」「TABLE」「ユーザインタフェース（カーソルやシステムカラーへの参照）」「音声再生」に関する指定が可能になっています．

1.1.3. HTML文書とスタイルシートファイルの連携

スタイルシートはテキストファイルとして作成し，保存しておきます．そのスタイルシートを利用するには，HTML文書中に「このスタイルシートを使う」といった指定を（LINK要素として）書き込めばよいのです．

WWWブラウザは，読み込んだHTML文書にスタイルシートファイルが指定されていると，自動的にそのシートを読み込みます．そして，それにしたがって表示を調整するのです．

リスト1●CSSファイルの例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>HTML4.0(W3C Recommendation 18-Dec-1997)解説</TITLE>
<LINK REL="STYLESHEET" TYPE="text/css" HREF="normal.css">
<LINK REL="STYLESHEET" TITLE="fancy" TYPE="text/css" HREF="fancy.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="compact" TYPE="text/css" HREF="small.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="compact" TYPE="text/css" HREF="mini.css">
</HEAD>
```

1.1.4. 簡単なスタイルシート例

1)見出しを調整

Netscape NavigatorやInternet Explorerの標準的な設定では，H1～H6の要素で指定される「見出しのレベル」が下がるにつれて，フォントサイズは順に小さくなります（**図1-1**）．

このH1などは通常，<H1>などと「<>」で囲まれたタグと呼ばれる形式で使用されています．詳しい用語の説明は2章で行いますが，「タグ」とは<H1>や</H1>といったもので，それに挟まれた内容は「要素」と呼びます．

図 1-1. WWW ブラウザ標準の「見出し」

```
<H1>1.ももたろうを考える</H1>
          <H2>1.1.出発まで</H2>
                    <H3>1.1.1.はじまりはじまり</H3>
                    <H3>1.1.2.桃太郎の誕生</H3>
```

1. ももたろうを考える

1.1. 出発まで

1.1.1. はじまりはじまり

1.1.2. 桃太郎の誕生

これを，スタイルシートを用いて**図 1-2** のように変更してみましょう．

図 1-2. スタイルシートで調整した「見出し」

```
H1{ font-size: xx-large;}/*H1 をずば抜けて大きく*/
H2, H3{ font-size: large;}/*H2 と H3 は同じ大きさに*/
H2{ border-bottom: 1pt solid}/*H2 の下にボーダーラインを*/
H3{ margin-left: 3em;}/*H3 は左に余白を*/
/*「em」は「一文字分」のサイズ*/
```

1. ももたろうを考える

1.1. 出発まで

1.1.1. はじまりはじまり

1.1.2. 桃太郎の誕生

このように，ほんの数行のスタイルシートでも，見栄えを大幅に変更できるのです．

2)本文を調整

Netscape Navigator や Internet Explorer の標準的な設定では，段落（P 要素）の開始/終了は 1 行分の余白で表現され，行間はぴっちり詰まっています．また，強調部分（STRONG 要素）は太字で表現されます（**図 1-3**）．

図 1-3. WWW ブラウザ標準の「段落」

```
<H3>1.1.1.はじまりはじまり</H3>
<P>桃太郎のお話は，おとぎ話によくあるように，「むかしむかし，あるところに」で始まります．
おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます．</P>
<P>ところで，最近の人は「柴」をご存じでしょうか．柴とは無関係の生活をしているため，
<STRONG>「芝刈り」<STRONG>だと誤解している人もいるかもしれません．</P>
```

1.1.1.はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「**芝刈り**」 だと誤解している人もいるかもしれません。

それを，**図 1-4** のように変更してみましょう．

図 1-4. スタイルシートで調整した「段落」

```
P{
    text-indent: 1em; /* 行頭に1文字分の字下げ */
    line-height: 1.5em; /* 行幅は1.5文字分 */
    margin-bottom: 0em; /* 次の段落との余白は取らない */
}
STRONG{     /* 白黒反転 */
    color: white;
    background: black;
}
```

1.1.1.はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。
ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「芝刈り」だと誤解している人もいるかもしれません。

このように，ちょっとした指定でも，文章の印象を大きく変えられるのです．

1.1.5. どんなことが可能か?

これまでの説明で分かるように，スタイルシートを**ほんの数行**の記述するだけで，文書全体の見栄えを**大きく**変更できます．また，用意されているプロパティを利用すれば，ほぼ自在に見栄えを調整できます．プロパティを複雑に絡み合わせることによって，市販の書籍に匹敵するようなスタイル調整が可能です．また，WWWでは色や画像を気軽に指定できるので，いろいろな表現を手軽に楽しむこともできます（**図1-5**）.

図 1-5. スタイルシート適用例

このように多機能なスタイルシートですが，以下に示すような特殊な表示効果のプロパティは用意されていません.

- 段組，フレーム
- 文字の回転
- 色のグラデーション制御

スタイルシートは必ずしも万能では**ありません**．たとえば，スタイルシートで絵は書けません．スタイルシートを活用するには，まずスタイルシートで何ができるのか，何ができないのかを知り，その上でそれを活かすスタイル調整を考えるべきでしょう.

1-2 HTMLにまつわる誤解と混乱の歴史

スタイルシートの活用を考える前に，ちょっと歴史的な経緯を確認したいと思います．

1.2.1. 誤解？

これまでにHTMLの解説などで「見出しレベルの違いは，文字サイズの違いに相当する」という説明を見聞きしたことはないでしょうか？　また，「HTMLの段落P要素は，一行分の改行（余白）として表示される」という話はどうでしょう．「引用ブロックであるBLOCKQUOTE要素は，本文よりも大きなマージンをとって表示される」というのも聞いたことがないでしょうか．

すでにお分かりのとおり，これらの表示はスタイルシートによってまったく異なるものに変更できます．では，いったいこの説明は何を示しているのでしょう？　**じつは**，特定のWWWブラウザにおける標準の表示を説明しているに過ぎません．

HTMLの仕様そのものは，各要素の意味合いや役割—見出しであるとか，強調語句であるとか—を決めているだけで，その表示に関しては（原則として）何も決めて**いません**．なぜなら，表示はスタイルシートの範疇であり，HTMLの範疇ではないからです．

では，HTMLの範疇とはいったい何なのでしょう．

1.2.2. HTMLの仕事の範疇

（1）SGMLの考え方

スタイル指定に関して，次のことを考えてみてください．

❶文字サイズを変えたから，その語句が見出しになったのか？
❷見出しだと決めたから，文字サイズを変えたのか？

通常は❶ではなく❷の方が多いでしょう．すなわち，スタイル指定よりも先に，まずは文書の構成を明らかにしておくのが普通だと考えられます．❶の場合でも，文字サイズを変えるということは，無意識にその部分を見出しにしようとする意思が伴うことが多いということです．

文書のテキストデータのうち「どこが見出しでどこが本文なのか」を明示するための仕組みとして，SGML（Standard Generalized Markup Language）があります．このSGMLでは，見出しならば

```
<見出し>スタイルシートを使おう！</見出し>
```

と，文章に「タグ」と呼ばれる記号をつける（マークアップする）のです．

このような見出しの指定さえしていれば，あとはスタイルシート側で「見出しはフォントサイズ20ポイントで，太字，ゴシック」などと指定するだけで見栄えを調整できます．なかなか賢いしくみだと思いませんか?

（2）HTML ⊂ SGML（HTMLはSGMLに含まれる）

SGMLの仕組みでは，次の2つの過程によって文書を記述します．

❶文書の様式（文書型定義）を定義する
❷様式にしたがって実際の文書を書く

文書型定義とは，「これから文書で指定するのは，“見出し”と“本文”と“強調語句”です」といったことを明示するものです．すなわち，文書の様式そのものに相当します．しかし，毎回毎回文書の様式を定義していては大変です．そこで，文書型定義は使いまわしがきくようになっています．

じつはHTML（HyperText Markup Language）の正体は，この❶の過程を省略したSGMLなのです．HTMLにおける文書型定義はすでに定義済みで，われわれはそれにしたがって文書を記述するのです（このようにSGMLを活用した文書記述体系を，専門用語で「SGMLアプリケーション」といいます）．

したがって，HTMLの仕事の範疇は文書の構成を明示することであり，各文書の見栄えに関しては（原則として）何も規定していません．その見栄えに関してはスタイルシートに一任されているのです．

初めからSGMLだったのか?

最初のHTMLおよびWWWの仕組みはCERNで考え出されました．そのときには，HTMLは厳密なSGMLアプリケーション**ではなく**，もう少しルーズな（?）ものでした．ルーズであるがゆえに，逆に文書記述のルールにあいまいな部分がありました（CREN：ヨーロッパの研究機関．CERNのTim Berners-Leeが90年にWWWという仕組みを発表しました．なお，彼は現在W3Cのディレクターとしても活躍しています）．

しかし，その後の混乱期を経て**W3Cが設立され**，W3Cの努力によって**HTML2.0以降**は明確なSGMLアプリケーションになりました．

実際にはWWWブラウザ開発会社などが独自拡張したHTMLも存在するのですが，“標準”のHTMLといえば，W3Cが推奨するもの（W3C Recommendation）を指しています．

1.2.3. 誤解と混乱の歴史

（1）スタイルシートの不在

ところで，WWWが誕生したのは90年，WWWブラウザMOSAICにより爆発的に普及し始めたのは93年，CSS1が発表されたのは96年です．つまり，HTML文書のスタイル指定は6年間（あるいは3年間）放置されていました．

実際には考慮されていたのですが，標準化には至りませんでした．

一般的な感覚からみれば，この期間はそれほど長くはないと言えます．しかし，驚異的な成長を見せたWWWの発達過程と並べてみれば，“大変長い時間が経った”と感じざるを得ません．

MOSAIC

98年にもなると，すでにMOSAICをご存じないひともいるかもしれません．MOSAICは世界で最初にインライン画像表示を実現したWWWブラウザで，瞬く間に各種OSに移植され，93年には**事実上の標準WWWブラウザ**となりました．その開発者マーク・アンドレセン（Marc Andreessen ）らはNCSAの一員でしたが，MOSAICの著作権についてNCSAともめた結果，NCSAを飛び出し会社を設立しました．それが現在のNetscape Communications Corporationです．

`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/`
(NCSA ： The National Center for Supercomputing Applications ：イリノイ大学の内部機関)

（2）予期せぬデファクト・スタンダード

CSS1が発表される前も，多くのWWW利用者がHTML文書の見栄えを調整したいと考えていました．しかし，はじめに市場の大部分を占めたMOSAICは，HTML文書に対するスタイル変更法を提供しませんでした．そのMOSIACが行うHTML文書の表示は，「見出しレベルの違いは文字サイズの違いで表現する」「段落の開始/終了は一行分の余白で表現する」「強調文字は太字で表現する」「引用ブロックは本文よりも大きなマージンを取る」といったものでした．その後任であるNetscape社のNetscape Navigator，Microsoft社のInternet Explorerも，MOSAICが規定したHTML文書の見栄え設計を受け継ぎ，やはりスタイルを固定的なものとして扱ってきました．

この時期に多くの人がWWWを利用するようになり，当然多くの人がHTML文書を記述しました．この段階ではまだスタイルシート仕様が確立されておらず，また「HTML文書の表示スタイルは，本来自在に変更できるものだ」と言及する人が少なかった（いなかった?）ため，いつしか，「MOSIACがそのように表示した」に過ぎなかった表示結果が，あたかもHTMLが定義する見栄え設定であるかのように理解されました．しかし，それは**誤解**なのです．

この誤解が大きな混乱を生みました．すなわち，「HTMLは文書の見栄えを指定するものであり，残念ながらその表現力は低い」というイメージが定着しはじめたのです．

（3）見栄えを直接指定する独自要素の導入

その誤解に拍車をかけたのが，Netscape社などWWWブラウザ開発会社が**独自に**採用した要素（タグ）です．各社は自分のWWWブラウザが他社のものより高機能であることを示すために，特殊効果を実現する要素を独自に導入しました．たとえば，FONT要素を用いれば，文書中の好きな文字のサイズや色を変更できます．BLINK要素を用いれば，文字列を点滅させられます．MARQUEE要素を用いれば，電光掲示板のように流れていく文字を簡単に表示できます．

独自要素を導入すること自体は，WWWをより楽しいものにするという意味で，悪いことではありません．しかし，これらの要素の多くは，文書構成ではなく「表示のしかた」を指定するものでした．このような独自要素が注目されるにつれ，「HTMLとは文書の表示結果をタグで指定するものだ」というイメージがより強まってしまいました．

いまや，「文字のサイズを変更するには，H1～H6タグを用います」「表示に大きなマージンをとるには，BLOCKQUOTEタグを用います」と書いてある市販の本を探すことすら難しくありません．しかしこれでは，HTMLの本来の考え方が完全にどこかへ吹っ飛んでしまっています．

HTMLの誤解と混乱の歴史のまとめ

「見出し」を例に，HTMLそのものに関する誤解と混乱の歴史をまとめます．

本来の「見出し」の表現：

HTML	「見出し」を見出しタグで囲って明示
スタイルシート	「見出し」を大きな文字に指定
表示結果	「見出し」は大きな文字で表示される

誤解と混乱の歴史：

・スタイルシートが確立していなかった．
・利用者は，何よりも文書の見栄えに興味があった．
・したがって，用意された表現の中から好みのものを探し，利用するしかなかった．
・本来の「見出し」などの意味を考えずに，HTMLのタグを見栄えの指定道具として考えるようになった．

誤解と混乱の結果：

「大きな文字にしたいときには，見出しタグを用いる」

こうなると，HTMLが文書構成を示すものだとはなかなか理解できません．しかも，BLINKやMARQUEEなど一部の独自拡張要素は文書構成とは何の関係もありません．こうして，HTML本来の仕事は忘れ去られ，誤解と混乱ばかりが蔓延するようになったのです．

1.2.4. 怪しいテクニック

そのような誤解の中で，マークアップを流用して見栄えを調整する「テクニック」が数多く生み出されました．もっとも，それは誤解ではなく，スタイル表現を支配したいと苦心した結果なのかもしれません．しかし，どちらにせよそれらがHTMLマークアップを混乱させているのです．

以下に，代表的な“テクニック”とその問題点を紹介します．

▶ 特定のソフトウェアの表示結果に依存（マークアップの無意味化）

・マージンをとるために，引用したわけでもないのにBLOCKQUOTE要素に指定する
・文字を大きくするために，見出しでなくても見出し要素（H1～H6要素）に指定する
・見出しを作成するために，見出し要素を用いずに，FONT要素を用いる
・大きな改行を得るために無意味にBR要素を挿入する

この“テクニック”に従っていると，HTMLマークアップの本来の目的である「見出しの明示」などが統一できません．「見出しとして指定されているのが見出し」という単純なはずのルールが守られないと，文書を読むときに見出しがわかりにくくなります．

また，このように書かれたHTML文書では，スタイルシートで見栄えを調整しようにも，文書構成が統一されていないために調整できなくなってしまいます．

▶ 拡張マークアップの多用（一般性の崩壊）

・「文字列を点滅させる」……………………………………BLINK
・「文字列を中央寄せ表示する」……………………………CENTER
・「画面を分割して複数のHTML文書を同時に表示する」……FRAME
・「HTML文書上に独立したパネルを作成する」……………LAYER
・「文字をスクロール表示する」……………………………MARQUEE

これらの要素を利用すれば，そのWWWブラウザを利用している人たちには美しい効果として解釈されるでしょう．しかし，それ以外のWWWブラウザを利用している人はどうなるのでしょう？　この方法では，一部の人にしか効果をアピールできません．

独自拡張が後にW3Cに採用されるケースもあります．たとえば，FONT・CENTER・TABLEはもともとはNetscape社の独自拡張要素でしたが，HTML3.2で採用されました．FRAMEはHTML4.0で採用されました．

▶ すべてを画像として表現

一般に，影付き文字や会社のロゴマークのような特殊な装飾つきの文字は，テキストではなく画像としてHTML文書に貼り込みます．この極端な例として，文章全体を画像にして公開しているケースに遭遇することがあります．つまり，独自のDTPソフトウェアなどを用いて整形した結果のスクリーンショットでホームページを構成するのです．

たしかに，これならばソフトウェアに依存せずに，表示を完全に支配できます．しかし，次のような欠点があります．

・データサイズが増大し，転送に時間がかかる
・文書内のテキストを検索するなどの文字処理が不可能
・画像を表示できない人には，まったく情報が伝えられない（文字情報のみを表示するWWWブラウザや音声読み上げ式WWWブラウザなどにとっては致命的）

▶ すべてをTABLEとして表現

TABLEの本来の意味は「表」ですが，表のコマ（CELL）の位置を「文字の位置」として利用し，細かなレイアウトを実現した文書も存在します．中でも，上下左右のマージンや，語句同士の位置関係を調整する効果を狙ったものが多く見られます．

この“テクニック”の最大の難点は，「指定が難しい」ことです．複雑なレイアウトを実現しようとすればするほど，HTML文書のソースは複雑になります．すると，専用ソフトウェア（高機能のWYSIWYGのHTMLエディタなど）を用いなければ思ったとおりの効果を得るのは難しいといえます．

もうひとつ難点があるとすれば，処理に必要な時間です．複雑なレイアウトのTABLEを調整するには，当然それなりの処理時間が必要です．結果として文書の表示にかかる時間が増大するでしょう．これはコンピュータの処理速度に依存する問題ですが，読者をイライラさせる可能性が高いことは間違いありません．

TABLEはHTML3.2から正式採用されているものの，対応していないWWWブラウザもいまだ活躍しているという事実もお伝えしておきます．

混乱から抜け出すために

1.3.1. 意識の転換

「どうしてマージンをとるためにBLOCKQUOTE（引用ブロック）なんていう名前の要素を使うのだろう．どうしてちょっと余白を取りたいだけなのに，テーブルだのセルだのを考えなければならないのだろう．HTMLって**難しい**．これじゃあ，“気軽にWWWで文書公開”なんて無理だなあ．」

こう考えてしまうのはもっともです．しかし，それは「本来HTMLにはできないこと」を無理やりHTMLで行おうとしていたのがまずかったのです．

HTMLでスタイルを調整しようとするのを**止めましょう**．HTML文書を本来の文書構成を明示するためのものに戻しましょう．そして，**スタイルシートを使いましょう**．

1.3.2. スタイルシートを使う利点

1.2.5.で紹介した怪しいテクニックを用いる場合とは異なり，スタイルシートを用いた見栄え調整には以下のような利点があります．このことを逆に考えてみると，HTMLで見栄えを指定することがいかに「無理のある仕事」であったのかを実感できると思います．

（1）シートの記述しやすさ

◆テキストファイル

CSS仕様のスタイルシートはテキストファイルとして記述されるため，スタイルシート作成のために特別なソフトウェアを用意する必要はありません．

◆直感的なキーワード指定

簡単なキーワードを記述するだけで，文字サイズや色・マージン・インデント・行幅などを自在に設定できます．マージンならば[margin: 3cm]，フォントサイズならば[font-size: 13pt]など，直感的な英単語を利用するため学習も容易です．また，画面サイズなどに合わせた相対指定も可能です．

◆指定効果の汎用性の確保

指定の効果は仕様書に明示されているため，どのソフトウェアに対しても同様の効果を期待できます．

◆データサイズの軽量化

“継承（inheritance）”という仕組みを上手に活用すれば，少しの指定でも全体のスタイルを調整できます．

（2）HTML文書との相性の良さ

◆HTML文書とスタイル指定の分離

HTML文書から独立した仕組みでスタイルを指定するため，HTML文書がすっきりと分かりやすいものになります．

◆スタイルシートの再利用

ひとつのスタイルシートを複数のHTML文書から利用することが可能です．また，スタイルシートの中に別の既存のスタイルシートを取り入れる（import）ことも可能です．

◆非対応ソフトウェアへの考慮

スタイル指定がHTML文書から分離しているため，対応していないWWWブラウザは，「スタイルシートとHTML文書」を受け取ったとしても，単にHTML文書だけを処理できます．したがって，スタイルシート非対応ソフトウェアを用いている人にも支障なく文書内容を伝達できます．

◆ 読者の都合にあわせたスタイル調整

カスケーディングという仕組みによって，読者独自のスタイルシートを持ち込み，表示を調整することも可能です．そもそも，読者はスタイルシート機能をオフにすることもできます．このように，スタイルシートを用いたHTML文書は，読み手の都合にあわせて柔軟に変化させられます．

◆スタイルシートの選択肢の提供

HTML文書そのものを書き直さなくても，別のスタイルシートに付け替えるだけで，見栄えを大きく変更できます．また，1つのHTML文書にあらかじめ複数のシートを指定しておき，読者に好みのシートを選択してもらうようなプレゼンテーションも可能です（**図1-6，1-7，1-8**）．

図 1-6. スタイルシートを用いなかった例

1. ももたろうを考える

1. 1. 出発まで

1. 1. 1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「芝刈り」だと誤解している人もいるかもしれません。

「柴刈り」は，暖を取るために，お風呂をたくために，あるいは他の目的のために枯れ枝を拾ってくることです。「柴」は特定の樹種ではなく，低木・雑木林全般を指して使われるのが普通です。

…

1. 1. 2. 桃太郎の誕生

桃から生まれたので，桃太郎。なんて簡単なネーミングでしょう。しかし，この簡明さこそが名前の本質を表わしています。

名前は，その人（あるいはモノ）の生涯の特性を端的に示しているのが普通です。あるいは，こうなって欲しいという親族の期待が込められています。「太郎」や「介」は男を意味しており，「男らしい男になってほしい」という願いがまざまざと伝わってくる…という次第です。

…

図 1-7. スタイルシートを適用した例 1

1. ももたろうを考える

1.1. 出発まで

1.1.1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「芝刈り」だと誤解している人もいるかもしれません。

「柴刈り」は，暖を取るために，お風呂をたくために，あるいは他の目的のために枯れ枝を拾ってくることです。「柴」は特定の樹種ではなく，低木・雑木林全般を指して使われるのが普通です。

…

図 1-8. スタイルシートを適用した例 2

1. ももたろうを考える

1.1. 出発まで

1.1.1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「芝刈り」だと誤解している人もいるかもしれません。

「柴刈り」は，暖を取るために，お風呂をたくために，あるいは他の目的のために枯れ枝を拾ってくることです。「柴」は特定の樹種ではなく，低木・雑木林全般を指して使われるのが普通です。

…

1.1.2. 桃太郎の誕生

1-4 本書の薦める学習手順

1.4.1. 本書の構成と学習手順

本書は，CSSを用いたHTML文書のスタイル調整を解説するものです．単にCSSの仕様を紹介するにとどまらず，スタイル指定の考え方にまで言及します．

しかし，スタイルシートを活用するには，HTML文書が文書構成を示すものとして書かれている必要があります（端的に言えば，HTML文書から見栄え指定を剥奪し，見栄えをスタイルシートに押し込む準備をしなければいけません．そうすることによって，HTML文書はより単純で明快なものになります）．そのためには，まずHTML本来の仕様を理解し，必要に応じて誤解を解かなければなりません．そこで，本書は以下のような構成になりました．

◆2章：tutorial of HTML as SGML

2章では，HTML文書の本来の役割－文書の構成を明示すること－に関して，いったんSGMLに立ち返って解説します．

◆3章：some more HTML

3章では，もう少しだけHTMLに関する取り決めを説明します．

◆4章：CSS syntax in detail

4章では，CSSによるスタイルシート記述の文法的な側面を解説します．

◆5章：CSS properties for visual media in detail

5章では，CSSが規定する個々のプロパティのうち，中心的に用いられるvisualメディア用のプロパティを解説します．

◆6章：road to mastering CSS

6章では，実際にスタイルシートを構築する上で重要になる概念や，無理の無いシートにするための「考え方」を紹介します．

◆7章：new feature in CSS2

7章では，新たにCSS2で導入された「印刷」「TABLE」「音声再生」などのためのプロパティを解説します．

なお，この章の並びは筆者の考える望ましい学習ルートに過ぎず，絶対ではありません．興味に応じて，お好きな章から読み始めても構いません．ただし，後の章は前の章の知識を前提にしていますので，分からない単語や概念が現れましたら，必要に応じて前の章をお読みください．

1.4.2. 巻末資料について

巻末に，HTML4.0とCSS2のクイックリファレンスを資料として収録しました．HTML4.0やCSSのすべてのキーワードを暗記する必要はありません．クイックリファレンスを机の脇に置き，活用してくださることを願います．

他にも，98年5月段階での各種WWWブラウザのスタイルシートサポート状況を収録しましたので，ぜひご利用ください．

1.4.3. 動作確認について

本書は，特定の実装結果（ソフトウェアの動作結果）ではなく，仕様をもとにHTMLやCSSを解説します．しかし，現実的には仕様と実装に**食い違い**があるものです．すなわち，仕様にはあるが機能しない概念，仕様とは異なった結果になるスタイル指定例などが存在します．

そこで，本書では代表的なWWWブラウザであるNetscape Navigator4.0とInternet Explorer4.0における実装と仕様との食い違いを必要に応じて報告します．ただし，筆者の都合で，Windows95におけるテストしか行えませんでした．Macや各種UNIXをお使いの方，あるいは他のWWWブラウザを用いている方はあらかじめその旨をご了承くださるようお願い申し上げます．

また，WWWブラウザの厳密なバージョンを書けば，Netscape Navigator4.04，Internet Explorer4.00です．執筆段階での最新バージョンはNetscape Nagivator4.05，Internet Explorer4.01ですが，これらはマイナーチェンジ（であるはず）なので，大きな違いはないものとさせていただきます．

1.4.4. WWWによるサポートと例ソース公開について

本書の6章で紹介するスタイル例は，すべてWWWで入手可能です．

```
http://www.gihyo.co.jp/css/
```

を参照してください．

tutorial of HTML as SGML

２章では，HTML 文書の本来の役割－文書の構成を明示すること－に関して，いったん SGML に立ち返って解説します．すでに HTML に関して知識のある方は読み飛ばしても構いませんが，知識を整理し，場合によっては誤解を解くために，ぜひ２章をお読みください．

2-1 SGML in brief

HTMLの概念を正しく把握するためには，SGMLに関する基礎知識が有用です．この節では，SGMLに関する基礎知識を「名刺」を題材に説明します．

2.1.1. 文書型定義と文書インスタンス

(1)「名刺」と「名刺の定義」

名刺とは，個人の名前や連絡先を印刷した簡単な文書です．それぞれに印刷されている文面は違いますが，**そのフォーマットはだいたい同じです**．細かい違いを思い切って無視してしまえば，『名刺とは，肩書きの紹介，名前の紹介，連絡先の紹介などで構成されているもの』と定義できるでしょう（**図2-1**）．

SGML的には，「名刺」を説明するには2つの段階を踏みます．すなわち，「名刺の定義」と「その定義にしたがって書かれた個々の名刺」をもって「名刺」を説明します．「名刺の定義」は「肩書き，名前，連絡先からなるもの」であり，「その定義に従った個々の文書」を「名刺」と呼ぶのです．あえて専門用語を用いれば，前者を「**文書型定義**（Document Type Definition：DTD）」といい，後者を「**文書インスタンス**（Instance：実例）」もしくは単に「文書」と呼びます．

図2-1.「名刺」と「名刺の定義」

○○商事営業部
円山幸雄
tel 052-123-4567

所属
姓名
連絡先

△△美術館館長
平山稔
tel 03-9876-5432
fax 03-9876-5431

文書型定義は，そのまま「文書の書き方の規則」になっています．おかしなたとえですが，銀行や役所で初めて必要書類を書く場合でも，**備え付けの「記入例」を見れば**正しく記入できるのと同様に，まだ「名刺」を作ったことのない人でも，**「名刺の文書型定義」を見れば，**必ず正しい様式で記述できるのです．

SGMLの考え方では，まず誰かが「文書型定義」を作成し，皆がそのルールにしたがって文書インスタンスを記述します．こうすることによって，「誰が書いても」同じルールにしたがって整えられた文書を記述できます．

ということは，すべての人が（少なくとも）**文書型定義の読み方を理解している必要**があります．

以降では，実際に「名刺」の文書型定義を作業する課程と，文書型定義にしたがって文書インスタンスを書く課程を紹介することで，SGMLに関する理解を深めたいと思います．

事実として，HTMLの仕様書でも，前半で文書型定義の読み方を紹介しています．

（2）「文書型定義」が定める用件

文書型定義は，次の用件を定義するものです．

- **どんな構成要素があるのか**
- **要素の出現順序，出現回数**
- **要素の親子関係**

文書型定義では，まずその文書を構成するパーツ（要素：Element）にどんなものがあるのかを明示しなければいけません．「名刺」の例では，「肩書き」「姓名」「連絡先」が構成要素です．

加えて，その各要素がどのような順番で何回出現するのかを定義すれば，文書型定義はほぼ完成です．ここでは，出現順序は「肩書き」「姓名」「連絡先」の順だと定義し，その出現回数は「肩書き」は0以上（1つも肩書きのない人もいるし，複数の肩書きをもつ人もいる），「姓名」は1（姓名のない名刺は困る），「連絡先」は1以上（最低でも1つは連絡先が必要）である，と定義します．

この場合は，文書型定義は次のように書かれます．

```
<!ELEMENT 名刺 - - (肩書き*, 姓名, 連絡先+) >
<!-- [解説]
    順序：肩書き、姓名、連絡先の順
    回数：0以上(*)、1、1以上(+)
-->
```

<!--から-->の間は，文書型定義における“コメント”です．

ところで，SGMLでは，ある要素は別の要素の集合として定義されています．ある要素を「親要素」と見た場合，それを構成している下位の要素を「子要素」と呼びます．さて，さきほどの文書型定義は最上位の要素「名刺」を定義したに過ぎません．続いてその子要素である「肩書き」「姓名」「連絡先」の文書型定義を記述しましょう．

```
<!ELEMENT 肩書き - - (#PCDATA) >
<!ELEMENT 姓名 - - (#PCDATA) >
<!ELEMENT 連絡先 - - (#PCDATA) >
```

ここで子要素として現れた「#PCDATA」は，SGML用語で「普通の文字」をさします．端的には，#PCDATAは目に見える（表示される）文字そのものです．全ての要素の子（孫）要素として#PCDATAが現れたところで，その文書の定義は終了です．すなわち，#PCDATAが現れてようやく本当に文書が記述できるわけです．

以上で「名刺」の文書型定義は**完成**しました（**図2-2**）．必要な知識はこれだけです．

図2-2.「名刺」の文書型定義（nomal）

```
<!--文書型定義「名刺」(http://www.meisi.com/dtd/nomal.dtd)-->
<!ELEMENT 名刺 - - (肩書き*, 姓名, 連絡先+) >
<!-- [解説]
    順序：肩書き，姓名，連絡先の順
    回数：0以上(*)，1，1以上(+)
-->
<!ELEMENT 肩書き - - (#PCDATA) >
<!ELEMENT 姓名 - - (#PCDATA) >
<!ELEMENT 連絡先 - - (#PCDATA) >
```

厳密には，各要素のとりうる“属性（Attribute）”に関しても定義が必要です．本書はSGMLそのものの解説書ではありませんので，あえて省略します．

（3）文書インスタンスの書き方

それでは，実際に文書型定義にしたがって「名刺」インスタンスを記述してみましょう．

文書インスタンスでは，タグ（tag）と呼ばれる記号を用いて要素を表現します．すなわち，「開始タグ<要素名>」と「終了タグ</要素名>」で挟むことで，その要素が記述されている範囲を特定するのです．

```
<要素名> 要素の内容 </要素名>
```

図2-2の**文書型定義によれば**，もっとも最上位の要素は「名刺」です．したがって，文書インスタンスの最も外側は次のようになります．

```
<名刺>
</名刺>
```

つづいて，**文書型定義**で「名刺」の子要素について**調べると**，「肩書き」（0回以上），「姓名」（1回），「連絡先」（1回以上）を書くことになっています．その定義にしたがって，「名刺」の内容としてこれらを記述しましょう（ここでは，すべての子要素を1回ずつ書いてみました）．

```
<名刺>
    <肩書き> </肩書き>
    <姓名> </姓名>
    <連絡先> </連絡先>
</名刺>
```

「肩書き」「姓名」「連絡先」の子要素は，#PCDATAすなわち普通の文字です．そこに実際の肩書きなどを書き入れれば，最終的な「名刺」インスタンスの**でき上がり**です（**図2-3**）．

図2-3．SGML文書インスタンスとしての「名刺」

○○商事営業部
円山幸雄
tel 052-123-4567

```
<名刺>
    <肩書き>○○商事営業部</肩書き>
    <姓名>円山幸雄</姓名>
    <連絡先>tel 052-123-4567</連絡先>
</名刺>
```

△△美術館館長
平山稔
tel 03-9876-5432
fax 03-9876-5431

```
<名刺>
    <肩書き>△△美術館館長</肩書き>
    <姓名>平山稔</姓名>
    <連絡先>tel 03-9876-5432</連絡先>
    <連絡先>fax 03-9876-5431</連絡先>
</名刺>
```

「タグ」≠「要素」

「タグ」は「要素の開始や終了を明示する記号」に過ぎません．しかし，「タグ」という用語を「要素そのもの」と混同して用いているケースをよく見かけます．たとえば，「スタイルシートでH1タグの見栄えを調整する」といった表現です．<H1>タグそのものは表示されないのですから，この表現は変です．ここはあくまでも「要素の見栄えを調整する」のです．

（4）HTMLの場合

以上のように，SGMLでは，文書を書くためにまず文書型定義を書き，その定義にしたがって文書インスタンスを書く必要がありました．一方HTMLは，はじめから文書型定義が定義されています．したがって，利用者は文書型定義に関して悩む必要はありません．ただ単純に，**与えられた文書型定義を理解し**，それにしたがって文書インスタンスを記述すればよいのです．

本書のHTML解説では，W3Cの定める正規の文書型定義を整理・単純化したものを用います．これを用いれば，すぐにHTML文書インスタンスを記述できるようになるでしょう．

SGML宣言について

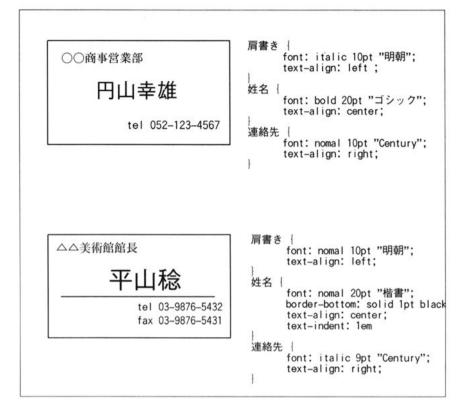
本当にSGMLアプリケーションを構築するには，文書型定義とともに，SGML宣言と呼ばれる一連の宣言書が必要になります．SGML宣言は，開始タグの記号は何にするのか，アルファベットの大文字小文字は区別するのか，などの取り決めを担当しています．とはいえ，多くの場合SGML宣言は標準指定されているものを用い，独自の修正（開始タグを「#+ +#」，終了タグを「#- +#」とする，など）は行いません．そのため，本書ではSGML宣言の解説は省略します．

2.1.2. スタイルシートの位置づけ

（1）スタイルシートの仕事

ところで，文書型定義だけでは，文書の性格は分かっても最終的な印刷結果は分かりません．見栄えの指定は，スタイルシートという別の仕組みで行います（SGMLとスタイルシートはまったく独立した仕組みです．原理的にはスタイルシート以外のスタイル指定方法を採用しても構いませんが，ここでは簡単のためにスタイルシートだけを紹介します）．

スタイルシートは，文書型定義で取り上げた要素ひとつひとつに関して，フォントサイズや色などを指定していくものです．スタイルシートは文書インスタンスごとに切り替えられるので，文書型定義が同じでも，異なった見栄えの文書を作成することが可能です（**図2-4**）．

図2-4. スタイルシートの例

(2) HTMLの場合

以上のように，SGMLでは文書作成者が別途スタイルシートを記述しない限り，表示に関しては完全に不明のままです．一方，W3CがHTML用に推奨するCascading Style Sheet（CSS）の仕組みでは，「WWWブラウザ」「文書作成者」「読者」の3者がスタイルシートを持ちよることを前提にしています．すなわち，文書作成者がスタイルシートを省略してもWWWブラウザが適切に文書を整形して表示してくれます．もちろん，スタイルシートを用いれば，自在に見栄えを調整できます．

2.1.3. バージョン違いの表現

(1) バージョン違いとは

図2-2で「名刺」の文書型定義を提示しましたが，それは名刺の定義の**一例**に過ぎません．現実的には**さまざまな種類の名刺の定義**が考えられます．たとえば，「肩書きは必ず1つ以上記述しなければならない」「連絡先はTELかFAXのどちらかでなければならない」などのバリエーションが考えられます．そのような違いを「バージョン違い」といいます．

異なるバージョンを表現するには，**異なる文書型定義**が必要です．**図2-2**の文書型定義をnomal.dtdとし，ここでは新たにvariation.dtdを記述してみましょう（**図2-5**）．

図 2-5.「名刺」の文書型定義のバージョン違い

```
<!--文書型定義「名刺」(http://www.meisi.com/dtd/variation.dtd) -->
<!ELEMENT 名刺 - - (肩書き+, 姓名, 連絡先) >
<!-- [解説]
    順序：肩書き，姓名，連絡先の順
    回数：1以上(+), 1, 1
-->
<!ELEMENT 肩書き - - (#PCDATA) >
<!ELEMENT 姓名 - - (#PCDATA) >
<!ELEMENT 連絡先 - - (TEL | FAX)+ >
<!-- [解説]
    順序：TEL あるいは FAX どちらか一方，の1回以上の繰り返し
    回数：1，の1回以上の繰り返し
-->
<!ELEMENT TEL - - (#PCDATA) >
<!ELEMENT FAX - - (#PCDATA) >
```

文書型定義 variation.dtd にしたがってインスタンスを書いてみましょう（**図 2-6**）.

図 2-6. variation.dtd による文書インスタンス

```
<名刺>
    <肩書き>△△美術館館長</肩書き>
    <姓名>平山稔</姓名>
    <連絡先>
        <TEL>tel 03-9876-5432</TEL>
        <FAX>fax 03-9876-5431</FAX>
    </連絡先>
</名刺>
```

△△美術館館長

平山稔

tel 03-9876-5432
fax 03-9876-5431

（2）DOCTYPE 宣言

図 2-6 の文書インスタンスを解釈するためには，どの文書型定義を読めばよいでしょうか．もちろん，variation.dtd です．誤って nomal.dtd で解釈してしまうと，要素「TEL」「FAX」が理解できずに困ることになります．

文書を解釈する上で読者を混乱させるのは望ましくありません．その混乱を避けるために，おのおのの文書インスタンスの最初の1行に，それが**どの文書型定義にしたがって書かれているのか**を明示する宣言－DOCTYPE 宣言（文書型宣言）－を記述しましょう．

DOCTYPE 宣言は，一般に次の形式を取ります．

```
<!DOCTYPE 定義名 SYSTEM あるいは PUBLIC "DTD ファイルの位置">
```

なお，SYSTEM は非公開（ローカル）DTD（文書型定義）を，PUBLIC は公開 DTD を意味します．これまでの文書インスタンスに DOCTYPE 宣言を加えると**図 2-7** のようになります．

図2-7. DOCTYPE宣言付きの文書インスタンス

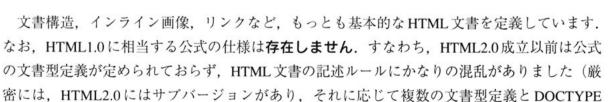
```
<!DOCTYPE 名刺 SYSTEM "http://www.meisi.com/dtd/nomal.dtd">
<名刺>
    <肩書き>△△美術館館長</肩書き>
    <姓名>平山稔</姓名>
    <連絡先>tel 03-9876-5432</連絡先>
    <連絡先>fax 03-9876-5431</連絡先>
</名刺>

<!DOCTYPE 名刺 SYSTEM "http://www.meisi.com/dtd/variation.dtd">
<名刺>
    <肩書き>△△美術館館長</肩書き>
    <姓名>平山稔</姓名>
    <連絡先>
        <TEL>tel 03-9876-5432</TEL>
        <FAX>fax 03-9876-5431</FAX>
    </連絡先>
</名刺>
```

厳密には，バージョン違いの存在にかかわらず，SGMLの文書インスタンスにはDOCTYPE宣言の記述が「必須」です．DOCTYPE宣言がなければ，読者はそもそも「名刺」なのか「HTML」なのかを判断できません．HTML文書を書く上でも，同様にDOCTYPE宣言が必須です．

（3）HTMLの場合

98年3月現在，公式に認められているHTMLのバージョンには2.0，3.2，4.0があります．

※HTML2.0：【95年12月，RFC1866】

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
あるいは
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

文書構造，インライン画像，リンクなど，もっとも基本的なHTML文書を定義しています．なお，HTML1.0に相当する公式の仕様は**存在しません**．すなわち，HTML2.0成立以前は公式の文書型定義が定められておらず，HTML文書の記述ルールにかなりの混乱がありました（厳密には，HTML2.0にはサブバージョンがあり，それに応じて複数の文書型定義とDOCTYPE宣言が存在します．詳しくはRFC1866をご参照ください）．

※ HTML3.2：【97年1月，W3C Recommendation】

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
あるいは
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">

HTML3.2は，Netscape社などの独自仕様の要素（TABLE，FONTなど）や独自属性（BGCOLOR，ALIGNなど）をHTML2.0に追加したものだとおおざっぱに説明できます．

なお，HTML3.2よりも前に，HTML2.0を大幅に拡張するHTML3.0が検討されたのですが，そのギャップの大きさ(?)のために3.0は破棄されました．

※ HTML4.0【97年12月，W3C Recommendation】

HTML4.0-strict：
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
HTML4.0-transitional：
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
HTML4.0-frameset：
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">

HTML3.2ではFONT要素など，直接にスタイルを規定する要素が採用されました．しかし，SGML的な観点からいえば，文書構成要素は文書における意味（「肩書き」「連絡先」など）に基づいて定義されるべきであり，FONT要素などは**望ましくないもの**です．しかし，すでにデファクト・スタンダードとして広まっている以上，これらを無視するわけにはいきません．

そこで，HTML4.0ではStrict，Transitional，Framesetのサブバージョンが制定されました．

StrictはFONT要素などのスタイル規定系要素・属性を破棄した厳密なバージョン，**Transitional**はスタイル規定系要素・属性を残したバージョンです．**Frameset**は毛色が異なり，フレーム表示のHTML文書に使われる特別な文書型定義です．なお，Transitional（過渡的）という名前が示すとおり，今後はStrictに移行することが勧案されています．スタイル指定はスタイルシートで行いましょう．

HTML4.0の新基軸としては，文書の国際化考慮・スタイルシートとの連携・マルチメディアオブジェクト・スクリプト・フレームの採用があげられます．また，テーブル，フォームの表現力がより豊かになりました．さらに，アクセス性に関する考慮（マウスを用いない読者，画像を表示しない読者にも文書内容を伝えるための考慮）のための属性が追加されています．

より具体的なアクセス性の確保に関しては，W3C内部にWeb Accessibility Initiative (WAI)という機関が設けられ，指針の草案を検討中です（http://www.w3.org/WAI/）．

国際化考慮について

厳密には，HTML2.0や3.2の仕様は西欧言語のアルファベットの文字コードを対象にしており，その他の言語圏--日本語やヘブライ語など--を考慮したものになっていません．すなわち，属性値などに日本語文字を用いてよいと言い切れないところがあるのです（注：もっとも，各種WWWブラウザの実装は別です．日本語対応をうたうWWWブラウザであれば，HTML2.0で日本語を用いていても難なく表示してのけるでしょう）．

HTML4.0では仕様として，はじめから国際化を考慮しています．すなわち，HTML4.0にしたがったHTML文書では，気兼ねなくHTML文書で日本語を使えます．

実は，HTML2.0を国際化対応に拡張した仕様も存在しました．

◆HTML2.X【97年1月，RFC2070】

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML i18n//EN">
```

i18n：internationalizationの略です．iで始まって，18文字あって，nで終わるからi18nです）

本書では国際化に関する具体的な解説は省略します．

（4）バージョン間の互換性について

HTMLのバージョンは上位互換では**ありません**．したがって，HTML2.0の仕様に則って書いていた文書インスタンスのDOCTYPE宣言を4.0に付け替えても，HTML4.0の文書として通用するとは限りません．

また，新しいバージョンが絶対では**ありません**．すなわち，HTML4.0でFONT要素が破棄されたからといって，HTML3.2として書かれた文書でFONT要素を使えなくなったわけではありません（もっとも，あまり望ましくはありませんが）．

HTML文書作成者は，どのバージョンの文書型定義にしたがってHTML文書を記述しても構いません．ただし，必ず相当するDOCTYPE宣言をつけてください．

DOCTYPE 宣言に関する誤解

「DOCTYPE 宣言をつけるように」と声高に叫ばれ始めたのは，HTML3.2 が制定されたころです．しかし，説明が不十分な場合が多く，「HTML 文書を書く場合は，必ず，

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
```

と頭につけておくように」という説明すら存在しました．

実際には，**記述した HTML のバージョンに応じて**正しい DOCTYPE 宣言を記述しなければいけません．たとえば，HTML2.0 の仕様にしたがって書いた HTML 文書であれば，HTML2.0 の DOCTYPE 宣言を記述してください．

また，BLINK 要素や MARQUEE 要素は WWW ブラウザ開発会社の独自拡張要素です．これらを用いた HTML 文書は，公式の仕様にのっとった HTML 文書では**ありません**．したがって，上に挙げた DOCTYPE 宣言を記述するのは誤りです．どのような DOCTYPE 宣言を書くべきかは，Microsoft 社や Netscape 社に聞いてください ^^;)．

以上で，必要な予備知識はすべて解説しました．「らくらくカンタン」ではないものの，思ったよりは難しくなかったのではないでしょうか．

2-2 HTML as SGML in brief

続いて，HTMLの解説を，その文書定義をもとに行います．

2.2.1. HTMLの文書型定義（簡略版）

本書はHTML4.0-Strictをベースに，文書記述のコア部分だけを取り出した文書型定義を用いてHTMLを解説します．本書の指示にしたがって書かれたHTMLインスタンスは，必ずHTML4.0-Strictの仕様を満足しますので，4.0-StrictのDOCTYPE宣言を記述してかまいません．

▶HTML文書（HTML要素そのもの）は**HEAD要素とBODY要素**で構成されます．HEAD要素は前書き，BODY要素は本文に相当します．

▶HEAD要素は**TITLE要素**で構成されます．TITLE要素は文書のタイトルに相当し，#PCDATAです．

▶BODY要素は，1つ以上の**「ブロック系要素」**の繰り返しです．ブロック系要素は自然言語の「段落」に相当するもので，その要素の開始・終了で表示が改行されます．

▶「ブロック系要素」は，**#PCDATAか「インライン系要素」**の繰り返しです．インライン系要素は，ブロック内部の特別な単語（強調したい単語など）などを示唆する要素です．

▶「インライン系要素」は，**#PCDATAか「インライン系要素」**の繰り返しです．すなわち，インライン系要素の内側に別のインライン系要素を入れ込むことが可能です．

以上をまとめると，まず前書きがあって，続いて本文がきます．本文は段落の繰り返しです．段落は，普通の語句と強調したい語句などの繰り返しです．さて，その文書型定義は次のように単純なものです（図2-8）．

図 2-8. HTML4.0 の文書型定義（簡略版）

```
<!--HTML4.0のサブセット.
    インスタンスはHTML4.0-StrictとDOCTYPE宣言して構わない.
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
-->

<!-- 最上位の要素は「HTML」-->
<!ELEMENT HTML o o (HEAD, BODY) >
<!-- [解説]
    順序：HEAD, BODYの順     回数：1, 1
-->

<!ELEMENT HEAD o o (TITLE) >
<!ELEMENT BODY o o (%ブロック系要素;)+ >
<!--
    回数：1以上(+)
-->
<!-- この%付きの「名前」は,「要素のグループ」の名前であり,
       HTML文書中にはそのグループ内の要素を記述する.
       C言語の#defineマクロ展開に似ている.
       具体的な要素は, 本文中で紹介する.
 -->

<!ELEMENT TITLE - - (#PCDATA)>

<!ELEMENT %ブロック系要素; - - (%インライン系要素;)+ >
<!--
     回数：1以上(+)
 -->

<!ELEMENT %インライン系要素; - - (#PCDATA | %インライン系要素;)+ >
<!-- [解説]
     順序：#PCDATAあるいは%インライン系要素;のどちらか一方,
           の1回以上の繰り返し
     回数：1, の1回以上の繰り返し
 -->

<!--
     本当は「%ブロック系要素;」と「%インライン系要素;」の
     実体を定義する必要があるのだが, ここでは省略する.
 -->
```

SGMLでは，要素を再帰的（入れ子的）に定義できます．すなわち，「インライン系要素」の子要素に再び「インライン系要素」が現れても構いません．

2.2.2. ブロック（段落）を組み上げる

文書型定義によれば，HTML文書インスタンスは前書き（HEAD要素）と本文（BODY要素）に分けられます．そして，本文（BODY要素）はブロック系要素の繰り返しです．したがって，HTML文書を書くには，まずHEAD・BODY・「ブロック」を組み上げることになります．

```
<HTML>
    <HEAD>
        <TITLE> title</TITLE>
    </HEAD>
    <BODY>
    </BODY>
</HTML>
```

ブロックは「段落」に相当するようなもので，表示の約束として，その要素の開始と終了で「改行」されます（これは，HTML仕様が表示結果に踏み込んでいる数少ない例です．もっとも，HTML4.0では「一般的には改行される」というような表現に改められています．なお，実際に改行するかどうかはスタイルシートで再定義できます）．

基本のブロック系要素は「見出し（Heading）」と「段落（Paragraph）」の2種類です．「見出し」は見出しレベルに応じて<H1></H1>，<H2></H2>，…<H6></H6>の6種類の要素として記述します（ごくまれにH7要素があると紹介されることがありますが，W3Cの仕様では存在しませんのであしからず）．段落は<P></P>と記述します．

ここでブロック開始・終了以外における「改行」が気になるかもしれませんが，通常は文字列が画面の端に達したところで自動的に折り返されるので，「改行指定」を意識する必要はありません．

```
<!ELEMENT (H1|H2|H3|H4|H5|H6) - - (%インライン系要素;)+ >

<!ELEMENT P - o (%インライン系要素;)+ >
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>「おとぎ話を考える」</TITLE>
</HEAD>
<BODY>
<H1>1.ももたろうを考える</H1>
        <H2>1.1.出発まで</H2>
                <H3>1.1.1.はじまりはじまり</H3>
<P>桃太郎のお話は，おとぎ話によくあるように，「むかしむかし，あるところに」で始まります．
おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます．</P>
<P>ところで，最近の人は「柴」をご存じでしょうか．柴とは無関係の生活をしているため，
「芝刈り」だと誤解している人もいるかもしれません．</P>
                <H3>1.1.2.桃太郎の誕生</H3>
<P>桃から生まれたので，桃太郎．なんて簡単なネーミングでしょう．
しかし，この簡明さこそが名前の本質を表わしています．…</P>
</BODY>
</HTML>
```

ありがちな誤解

見出し要素に関する**もっと多い誤解**として，「Hx 要素は文字のサイズを指定する」というものがあります．また，一歩正解に近づいたものの間違っているのは「見出し文字のサイズを指定する」という説明です．正しくは，「見出しレベルを指定する」です．表示のサイズはスタイルシートで自在に変更できます．

P 要素に関するもっとも多い誤解は，「P は 2 行改行する」です．また，その誤解の源になっている誤解は「P には終了タグはなく，改行したいところに付ける」というものです．**じつは**，HTML2.0 成立以前には，P 要素には終了タグがありませんでした．でも，それは**昔のことです**．現在は，段落の開始と終了を指定するように変更されています．また，改行幅 ── 実際には「下の余白」 ── はスタイルシートで自在に変更できます．

「見出しレベル」とは

見出しには大見出し，中見出し，小見出しといったレベルの違いがあります．別の表現をすれば，部・章・節・項などの階層になります．本書では，「1，2，3」といった最上位の見出し（第1レベルの見出し）と，その中にある「1.1，1.2，1.3」といった下位の見出し（第2レベルの見出し），「1.1.1，1.1.2，1.1.3」といった更に下位の見出し（第3レベルの見出し）があります．

HTMLでは，第1レベルの見出しはH1要素で表現します．第2レベルはH2，第3はH3であり，順にH6まで用意されています．

```
<H1>1.ももたろうを考える</H1>
        <H2>1.1.出発まで</H2>
                <H3>1.1.1.はじまりはじまり</H3>
                <H3>1.1.2.桃太郎の誕生</H3>
        <H2>1.2.鬼が島へ</H2>
                <H3>1.2.1.旅の仲間</H3>
```

HTMLの文法には，見出しの順序や付け方の制限はありません．しかし，通常はレベル1の見出しの次にレベル3の見出しがくるのはおかしなことだと思われます．

基本的な文書作成に必要な知識は，以上ですべてです．本当はまだ別のブロック系要素を紹介したいのですが，それは3章にまわします．

2.2.3. インライン装飾をする

```
<!ELEMENT %インライン系要素 - - (#PCDATA | %インライン系要素;)+ >
```

（1）STRONG要素

インライン系要素は，ブロック内部の**文章の一部を装飾するための**要素です．まずは，その代表格であるSTRONG要素を紹介します．STRONG要素は，強調したい部分を明示するために用います．

例として，次のような段落を考えます．

```
<P>私の誕生日は2月24日です．ひろのみやさんの翌日です．</P>
```

もし，この文章で「2月24日」を強調したいのであれば，その部分を（ただのP要素ではなく）**STRONG要素として**設定します．

```
<P>私の誕生日は<STRONG>2月24日</STRONG>です．
```

```
ひろのみやさんの翌日です．</P>
```

こうすると，WWWブラウザや読者はこの部分を強調するべき語句（単語）だと理解できるようになります．

ほかにも強調したい語句（単語）があれば，それもSTRONG要素として記述しましょう．

```
<P>私の誕生日は<STRONG>2月24日</STRONG>です．
<STRONG>ひろのみや</STRONG>さんの翌日です．</P>
```

なお，すべてのブロック系要素の内部にインライン系要素を記述できます．すなわち，見出しの中にSTRONG要素を配置することも可能です．

```
<H1><STRONG>誕生日</STRONG>について</H1>

<P>私の誕生日は<STRONG>2月24日</STRONG>です．
<STRONG>ひろのみや</STRONG>さんの翌日です．</P>
```

文章を書いたときに心の中で「2月24日は強調したい」と思っても，**思っただけでは**読者もWWWブラウザもそれを理解することはできません．必ず**明示**してください．

(2) CLASS属性の設定

もし，それぞれのSTRONG要素で強調する理由が違うのならば，**その理由を明示**したほうが文章表現が豊かになります．HTMLの場合，そのような違いの表現をCLASS属性（Attribute）の指定で行います（「属性」に関する詳しい説明は3章で行いますが，さしあたって開始タグの中に<要素名 属性名1="値" 属性名2="値">などと記述するのだと思ってください．なお，終了タグには何も書きません）．

```
<P>私の誕生日は<STRONG CLASS="DATE">2月24日</STRONG>です．
<STRONG CLASS="NAME">ひろのみや</STRONG>さんの翌日です．</P>
```

CLASS属性は，その名のとおり，個々の要素の**所属するクラス（分類）**を示すものです．この例では，DATE（日付）クラスとNAME（名前）クラスを用いました．クラス名はあらかじめ用意されているわけではなく，文書記述者が任意に規定します．

ちなみに，スタイルシートを用いれば，要素の表示方法をクラス毎に設定できます．同じSTRONG要素でも，「DATE」は斜体，「NAME」は太字，というように表示結果を指定変えられるのです．したがって，適切にクラス属性を設定しておけば，より表現力豊かな文書になります．

```
<P>私の誕生日は<STRONG CLASS="DATE">2月24日</STRONG>です．
<STRONG CLASS="NAME">ひろのみや</STRONG>さんの翌日です．
弟の誕生日は<STRONG CLASS="DATE">4月28日</STRONG>で，
```

```
こちらは <STRONG CLASS="NAME">ひろひと天皇</STRONG>の前日. </P>

<P> しかも, “ににんがし” と “しにがはち” です.
嘘っぽいけど, 本当の誕生日です. </P>
```

CLASS 属性は，STRONG 要素だけでなく BODY 要素を構成するほぼすべての要素に指定できます．もちろん，P 要素や H1 要素にも指定可能です．

クラス名の付け方

クラス名は**「文章中での意味」に基づいて**付けることが重要です．もし，期待するスタイル効果に基づいたクラス名「RED」「BIG」などを採用すると，文書構成にとっては何の意味も無いばかりか，最終的にはHTML 内にスタイルを埋め込んだのと同じようにマークアップが混乱してしまいます．“大きくしたい” などの見栄え指定はスタイルシートに任せましょう．HTML 文書には，見栄えを変える根拠（人名だから，など）を記述しましょう．

クラス名に使える文字

クラス名として使える文字にとくに制限はありませんが，基本的には**半角英数字のみ**で書いてください．「!"#%&.」などの記号は他の用途で使われることが決まっているので，避けたほうが無難でしょう．また，日本語も避けましょう．

なお，クラス名は**スペースで区切って**複数記述することが可能です．すなわち，

```
<STRONG CLASS="SPECIAL DAY">2 月 24 日 </STRONG>
```

という記述は，この要素が「SPECIAL」クラスかつ「DAY」クラスに所属していることをあらわします．

逆に言えば，スペースは文法的に予約された記号なので，クラス名には使えません．

なお，アンダースコア（_）は使用可能です．次の例は，「SPECIAL_DAY」という 1 つのクラス名として解釈されます．

```
<STRONG CLASS="SPECIAL_DAY">2 月 24 日 </STRONG>
```

（3）その他のインライン系要素

もう少しインライン系要素を紹介します（**表 2-1**）．

表 2-1．さらなるインライン系要素

要素名	相当する意味
Q	引用語句（quotation）． 坂本竜馬曰く，<Q>「日本の夜明けは…」</Q>
CITE	引用もとの記事や本のタイトル． 先月の <CITE>「Software Magazine」</CITE> には…
SUB	下付き文字（subscript）． CO₂
SUP	上付き文字（superscript）． e = mc²

この中に望みの要素が用意されて**いない**場合は，SPAN 要素を用いて擬似的に表現してください．SPAN 要素は意味の決められていないインライン系要素で，CLASS 属性によってのみ意味を伝達します．ただしSPAN 要素は，スタイルシートで指定しない限り見栄えの変化を起こしません．

```
<H1><SPAN CLASS="number>1.</SPAN>はじめに</H1>
<H2><SPAN CLASS="number">1.1.</SPAN>背景</H2>
```

以上で，よく使われるインライン系要素はすべて紹介しました．なお，特殊な効果を生むインライン系要素としてIMG(インライン画像：Image)，A（アンカー：Anchor)，BR（改行：Line Break）があります．これらに関しては3章で解説します．

スタイル規定に関する要素に関して

すでにHTMLをご存じのかたは，フォントのサイズや色を指定するFONT要素を知っていると思います．**しかし，**HTML4.0-StrictではFONT要素は**破棄されました**．なぜなら，そのような指定はスタイルシートで行うべきだからです．HTML文書インスタンス内部には文書の論理的な構成のみを記述し，見栄えはスタイルシートに分離するようにしましょう．

なお，HTML4.0-StrictでもB要素（太字），I要素（斜体）などのスタイル規定系要素は採用されています．しかし，本書では**これらもスタイルシート側で指定すべきだ**として，あえて紹介しません．たとえば，太字にする理由が「強調」であればSTRONG要素を用いるべきですし，理由が「引用タイトル」であればCITE要素を用いるべきです．これはWAI（http://www.w3c.org/WAI/）も示唆している考え方です．

HTML文書を書く際には，次のように考えていただけると幸いです．

- **HTMLは，原則として文書構成を明示するものであり，スタイルを直に示唆するものは排除したほうがよい．**
- **現に，3.2で採用されたスタイル関連の要素はHTML4.0-Strictでは破棄された．**
- **例外的に，まだいくつかスタイル関連の要素が残っているが，使わないほうがよい．**

また，BLINK要素・MARQUEE要素などの「WWWブラウザ会社による独自拡張要素」の中には強力な見栄え効果を提供するものがありますが，これらの利用もお勧めできません．

スタイル規定に関する属性に関して

HTML3.2 では，いくつかのスタイル規定のための属性が採用されました．これらのうち，よく使われているものの，HTML4.0-Strict で破棄されたものを**表 2-2** に挙げます．

表 2-2．HTML4.0-Strict で破棄されたスタイル規定に関する属性

属性名	機能	対象要素
ALIGN	水平位置そろえ	H1 ～ H6，P，HR
ALIGN	フロート指定	IMG
ALIGN	垂直位置そろえ	IMG
BACKGROUND	背景画像	BODY
BGCOLOR	背景色	BODY
TEXT	一般文字色	BODY
LINK	未リンクのアンカー文字色	BODY
VLINK	既リンクのアンカー文字色	BODY
ALINK	指示最中のアンカー文字色	BODY
SIZE	HR 要素では，縦の幅	HR
WIDTH	HR 要素では，横の幅	HR
NOSHADE	HR 要素では，非立体表示	HR

「同じ ALIGN という属性名がこれだけ多岐にわたる機能に対して使いまわされている」という事実からも，この属性の混乱ぶりがうかがえるような気がします．

ちなみに，これらの機能はスタイルシートを用いれば，**より高度に**実現できます．たとえば，「H1 ～ H6 と P で異なる文字色にする」「リンク状態の違いを背景色とフォントファミリーの違いで表現する」などが可能です．

しかし，「スタイルシート非対応 WWW ブラウザのためにこれらの属性を残したい」かもしれません．そのような場合は，HTML3.2 か 4.0-Transitional の文書型定義にしたがってください．なお，本書の提示する文書型定義にこれらの属性を付け足したものは，4.0-Transitional の仕様を満足します．

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

2-3 Check Your HTML Document

誰が文章を書いてもほ，その中には何らかの間違いが潜んでいるものです．単なる打ち間違いもあれば，文法の誤解によるミスもあるかもしれません．WWWで公開する文書は人の目に触れる文章ですから，**公開前に**間違いを取り除く努力をするべきです．ここでは，HTMLのマークアップ上のミスと，それを探し出す文法チェッカーを紹介します．

2.3.1. どんなミスがありえるか

（1）単なる打ち間違い

▶STRONGをSTRINGと打ってしまっていた!

▶ 終了タグのつもりで開始タグを書いていた!

（2）親子関係の間違い

▶ インライン系要素を，ブロックを超えて記述していた!

不正なインライン系要素：

```
<STRONG>
<P>僕はビートルズが好きです．</P>
<P>しかし，ビートルズならばなんでもよいかといえば，そうでもありません．</P>
</STRONG>
```

正しいインライン系要素：

```
<P><STRONG>僕はビートルズが好きです．</STRONG></P>
<P><STRONG>しかし，ビートルズならばなんでもよいかといえば，
              そうでもありません．</STRONG></P>
```

▶TITLE要素をBODY要素の中に書いていた（正しくはHEAD要素の中に書く）

▶リストにおけるLI要素をリストコンテナ（UL，OL）の外に書いてしまった!（リストに関しては3章を参照）

（3）要素の入れ子の間違い

▶ 子要素が閉じる前に親要素が閉じてしまった

これは，インライン系要素を入れ子にしているときにありがちです．

不正な例：

```
<P>林業白書によれば，
<STRONG><Q>日本の森林率（面積割合）は67パーセント</STRONG>で，
世界でもトップレベルの森林保有国に位置づけられています</Q>．</P>
```

正しい入れ子構造：

```
<P>林業白書によれば，
<Q><STRONG>日本の森林率（面積割合）は67パーセント</STRONG>で，
世界でもトップレベルの森林保有国に位置づけられています</Q>．</P>
```

（4）順序，回数の間違い

▶BODY要素を何度も書いてしまった（HEAD，BODYは1度ずつしか書けません）

▶HEAD要素に直に文字（#PCDATA）を書いてしまった（HEADの子要素に#PCDATAはありません）

▶リスト項目（LI要素）のないリストコンテナ（UL，OL）を作ってしまった（リストコンテナの子要素として，LI要素はひとつ以上必要です）

HTMLの文書型定義では，順序・回数を制限される要素はごくわずかしかありません．そのせいか，順序・回数には制限があることを知らない人もいるようです．

（5）DOCTYPE宣言の付け間違い

▶BGCOLOR属性を使っているのに，HTML4.0-Strictだと宣言してしまった

▶BLINK要素など独自拡張要素を使っているのに，W3CのHTMLだと宣言してしまった

▶ そもそもDOCTYPE宣言を付け忘れている

2.3.2. 文法チェッカーの紹介

Netscape Navigator，Internet ExplorerなどのWWWブラウザは，HTML文書の文法が間違っていても「エラーだ」などと警告してはくれません．それどころか，ミスの存在を黙認して（?）何とか表示してくれます．すなわち，**よほど重大なミス**が存在しない限り，HTML文書は何とか表示されるものなのです（エラー処理の方法は，WWWブラウザによって異なります）．

しかし，「文法が間違ったHTML文書を書いてもよい」わけでは**ありません**．WWWといえ

ども人の目に触れるものとして文書を公開するのですから，**礼儀として**文法的なエラーは取り除いておくのが「義務」だと言えるでしょう．

HTML文書の文法をチェックするソフトウェアは複数存在しますので，これらを利用して間違いを見つけるとよいでしょう．ここでは，WWWで利用できるものを紹介します．

◆ Weblint/jweblint

`http://saturn.aichi-u.ac.jp/~mimasa/jweblint/`

Neil Bowers氏が作成したチェッカー．Masayasu Ishikawa 氏が日本語化したものを作成しています（Masayasu Ishikawa 氏はW3Cのメンバーです）．

Weblintは，文法チェッカーというよりも**作法チェッカー**です．すなわち，「望ましくないマークアップになっていないか」をチェックしてくれます．しかし，一部の文法的ミスを見逃してしまいます．

◆ Another HTML-lint

`http://ring.aist.go.jp/openlab/k16/htmllint/htmllint.html`

k16氏がWeblintに触発されて作成した文法チェッカー．Weblintとは異なり，文書型定義に基づいた**厳密な文法チェック**を行います．同時に作法チェックを行い，なんと**採点まで**してくれます．もっとも，得点はあくまでも参考ですので気をつけて．

Another HTMLlintの「チェック内容の解説」を読むと，HTMLの作法について大変勉強になります．1度目を通してみるとよいでしょう．

Some more HTML

3章では，もう少しだけHTMLに関する取り決めを説明します．しかし，スタイルシート記述にとっては本章の内容は「余談」です．すでにHTMLに詳しい方や早くCSSについて知りたい方は，読み飛ばしてもかまいません．

some more rules as SGML

3.1.1. タグ記述ルール

(1) 開始タグ・終了タグの省略できる要素

HTMLでは，基本的にはすべての要素において「開始タグ」と「終了タグ」を**明確に**記述しなければいけません．しかし，その開始や終了が明確である場合で，かつ文書型定義が許可している要素に限り，開始タグや終了タグの記述を省略できます．なお，文書型定義における要素名の直後の2つの記号が「省略の可・不可」を示しています（図3-1）．

2章で紹介した要素では，BODY要素とHEAD要素が「開始タグ」も「終了タグ」も省略可能です．また，P要素は「終了タグ」のみ省略可能です．逆に，それ以外の要素のタグはすべて明示する必要があります．

図3-1．文書型定義におけるタグ省略の可不可の提示

【注】oはアルファベットのオーで，omitの頭文字です．

文書型定義で「省略可」となっていても，無条件にタグを省略できるわけではありません．タグの省略が許されるのは，**要素の開始や終了が明確である場合**だけです．

▶ **開始タグが省略できるケース**

・**順序**から見てその要素の**開始が明確**なとき
・その親要素の開始が明確で，かつ**順序的に**その要素しかありえないとき
（HTML 要素の次には HEAD 要素しか書けないため，HTML 要素の開始タグの直後には HEAD 要素の開始タグがくることは明確）
・その直前の要素の終了が明確で，かつ順序的にその要素しかありえないとき
（HEAD 要素の終了タグがあれば，そこで BODY 要素が始まることは明確）

▶ **終了タグが省略できるケース**

・**順序的に**その要素の**終了が明確**なとき
・自分の**下位要素でない要素**の開始タグが現れるとき
（HEAD 要素には P や H1 は書けないので，これらが現われれば HEAD の終了は明確．もちろん，BODY が開始すれば HEAD の終了は明確）
・自分の**親要素の終了タグ**が現れるとき
（BODY 中の最後の P は，BODY の終了タグがあれば P の終了は明確）

このようなケースであれば，WWW ブラウザは機械的に「書かれていないけれど，本当はある」と判断して処理を続けられます．だからこそ省略が許されるのです．

したがって WWW ブラウザは，</P> のある文章もない文章も同じように表示するはずです．にもかかわらず，実際には表示結果が異なる場合があります．これは WWW ブラウザ側の実装ミスです．

「タグ」省略の指針

筆者は，あまりタグを**省略しないことをお勧め**します．たとえば，HTML・HEAD・BODY要素の開始タグ・終了タグを省略すると，文書構成が分かりにくくなるので望ましくないと思います．あえて使用してもよい「この場合だったら間違いなく省略してよい」ケースは，以下の2つのような場合です．

- P要素の終了タグは，その直後に**次のP要素の開始タグがくる**場合のみ，省略してもよい．
- LI要素の子要素が**インライン系要素だけ**の場合，そのLI要素の終了タグは省略してもよい．

```
<P>最近忙しいので，書きたいことがたまっている．たまりすぎて，忘れちゃいそう．
<P>忘れると困るので，タイトルだけでも書きとめておこう</P>
<UL>
    <LI>
        <P>Java関連</P>
        <UL>
            <LI>Java思想とJava言語，JavaVMの関係
            <LI>理想はどこまで実装されているか
            <LI>Sunは絶対正義なのか
        </UL>
    </LI>
    <LI>PRINCE音楽の構成学的な見地からの評価
    <LI>Netscape社だって必ずしも正義ではない
    <LI>IEを人に奨める際の注意事項
</UL>
```

もちろん，本当はもっと多くのケースで省略が可能です．しかし，筆者は「省略しない」ことをお勧めします（UL，LI要素に関しては3.2.1.を参照）．

（2）空要素

先ほどはタグを省略できる場合を紹介しましたが，実は「終了タグを**省略しなければならない**（書いてはいけない）」よう指定されている要素も存在します．それらは，後に紹介するIMG要素（画像）など，文字以外のものを表現する要素であり，「空要素（Empty Element）」と呼ばれます．空要素は開始タグのみを記述します．文書型定義では，空要素の子要素は「(EMPTY)」と書かれています（**図3-2**）．

厳密には，カッコは冗長です．しかし，本書は見やすさのためにカッコを追加しています．

図3-2. 文書型定義における空要素の提示

```
<!ELEMENT IMG - o (EMPTY) >
<!ELEMENT BR - o (EMPTY) >
```

（3）属性の記述

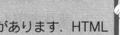
属性（Attribute）とは，要素に与えられる付加的な情報のことで，開始タグの中に記述します．終了タグには属性に関する情報は記述しません．属性値はクォーテーション・マークで囲みます．普通はダブル・クォーテーション（""）を用いますが，シングル・クォーテーション（''）でもかまいません．

<要素名 属性1="属性値" 属性2="属性値">文字列</要素名>

各要素が持っている属性は文書型定義で定義されています．

なお，HTML文書の場合，属性指定はあくまでも**付加的な**位置づけになっており，ほとんどの属性指定が省略可能です．また，いくつかの属性は見栄え調整のためのものであり，SGML的には望ましくありません．したがって，やみくもにすべての属性を覚えることは得策ではありません．

そこで本書では，省略不可の属性とCLASS属性など必要最低限のものに限って紹介します．

属性値の簡略記述法

たくさんのHTML文書を書いていると，属性値を囲む""付けがめんどうになることがあります．HTMLでは，以下の場合には省略してよいことになっています．

クォーテーションマークの省略：

値が「a-z, A-Z, 0-9, -（ハイフン），.（ピリオド）」のみからなる場合

省略した例，できない例：

```
<HR SIZE=5 WITDH="100%">
<P ALIGN=LEFT>～</P>
```

HTML4.0のSGML宣言が正しければ，4.0からは，さらに「-」「:」が含まれていても省略できるはずです．しかし，仕様書の解説では，互換性確保のためか，「-」「:」については言及されていません．

また，属性値が1つしか規定されていない属性（HRのNOSHADE属性など）に関しては，属性名の記述を省略できます（4.0までのHTMLに限って言えば，このような場合は属性名と属性値が同一にされており，見た目には属性名を省略したのか属性値を省略したのか区別できません）．

```
<HR NOSHADE>
```

現実的には，「省略するのが標準」と考えられるようです．一部のWWWブラウザは，“正式”な記述であるはずの<HR NOSHADE="NOSHADE">を「エラー」とするかもしれません．

なお，このコラムで紹介したSIZE，WIDTH，ALIGN，NOSHADE属性はすべて見栄え調整のための属性です．HTML3.2で採用されたものの，HTML4.0-Strictでは破棄されました．

（4）大文字・小文字および全角半角

タグ内部の要素名・属性名を記述するアルファベットの大文字，小文字は区別されません．ただし，必ず半角英数字を用いなければいけません．記号「<」「</」「>」「=」「""」「'」および空白に関しても半角英数字を用いてください．

実際の文章 ― すなわち #PCDATA ― 中に，日本語の空白として「全角の空白」を用いるのは大丈夫です．ここで禁止したのは，タグ内部など，HTML が定める記号を書く部分における使用です．

3.1.2. 実体参照に関して

記号「<」「>」「</」はタグを書くために予約されているため，HTML 文書のソースには（タグ以外の用途では）記述できません．ほかにもいくつか記述できない記号があるのですが，これらの記号であっても，「実体参照（entity reference）」と呼ばれる一種の別名を用いれば記述できます（C 言語の #define マクロ展開に似ています）．

実体参照は「&（アンド）」に始まり，「実体名」を書き，「;（セミコロン）」に終わります．なお，実体名は**大文字小文字が区別される**ので注意が必要です．

実体参照も文書型定義で定義されています（**表 3-1**）．

表 3-1. よく使われる実体参照

記述	表示	備考
&	&	ampersand の略
<	<	less than の略
>	>	greater than の略
"	"	quotation マークの略

「/」は直接記述できます．ただし，「</」は終了タグとしてしか記述できません．

3.1.3. 区切り文字（スペース，タブ，改行）に関して

HTML 文書内部では，単語の区切り文字として**スペース・タブおよび改行**を利用します（ただし，日本語文字の空白は区切り文字ではありません）．

区切り文字をソース中に何個**連続**して記述しても，読解時にはひとつの区切りとしてのみ処理されます．すなわち，次の2つのソースは同一の内容を表現しており，表示結果は同じになります．

```
<H1>1.ももたろうを考える</H1>
<H2>1.1.出発まで</H2>
<H3>1.1.1.はじまりはじまり</H3>
<P>桃太郎のお話は，おとぎ話によくあるように,「むかしむかし，あるところに」で始まります.
おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます．</P>
```

```
<H1>1.ももたろうを考える</H1>
    <H2>1.1.出発まで</H2>
        <H3>1.1.1.はじまりはじまり</H3>

        <P>桃太郎のお話は，おとぎ話によくあるように，
        「むかしむかし，あるところに」で始まります.
        おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます．</P>
```

HTML文書のソースでは，ソースとして見やすいように（表示に気がねせず）空白や改行を挿入してかまいません.

3.1.4. コメントアウト

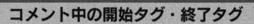
HTML文書のソースを読む人（ときには自分自身）のために，コメントを“WWWブラウザが表示できない形で”記述したいことがたまにあります．しかし，HTMLの仕組み自体にはコメント様式が用意されておらず，SGMLで文書型宣言を書く際のコメントアウトの書式を流用しています.

文書型定義は「<!」と「>」の間に書かれます，この中の「--」（ハイフン2つ）「--」で囲まれた部分は，コメントアウトされたものとして通常の処理から除外されます.

```
<!-- 次のリンクは，作成者に連絡して許可をもらった-->
<P><A HREF="http://www.foo.bar.jp/people/ks/">すみけんリソース!</A></P>
```

1つの「<!」と「>」の間に複数のコメントを記述することもできますが，コメントはあくまでも「--」が対になっていなければいけません（コメント中にはハイフンを記述しないのが最も安全でしょう）.

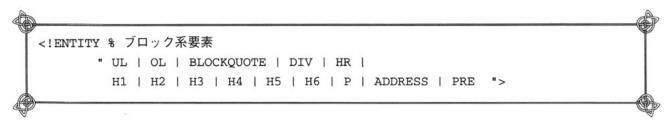
```
正しいコメントアウトの様式：
    <!-- リンク --   -- 作成者に連絡して許可をもらった-->
不正なコメントアウトの様式：
    <!-- リンク   -- 作成者に連絡して許可をもらった-->
```

コメント中の開始タグ・終了タグ

一部のWWWブラウザは，コメント中に終了タグがあるとそこでコメントが終了したと勘違いする，といわれています．なお，Netscape Navigator4.0とInternet Explorer4.0は正しくコメントアウトを解釈しました．

```
一部のwwwブラウザは，このコメントの解釈を失敗する：
<!--
      次のリンクは，まだ未許可のためコメントアウトしておく．
      <P><A HREF="http://www.foo.bar.jp/people/ks/">
      すみけんリソース!</A></P>
-->
```

some more ブロック系要素

この節では，もう少しだけブロック系要素を紹介します．なお，本本書の文書型定義はHTML4.0Structのサブセットであることをお忘れなく．

```
<!ENTITY % ブロック系要素
        " UL | OL | BLOCKQUOTE | DIV | HR |
          H1 | H2 | H3 | H4 | H5 | H6 | P | ADDRESS | PRE  ">
```

3.2.1. ブロックを含みうるブロック系要素

ここでは，子要素としてブロック系要素を含む要素を紹介します．これらの要素によって，より複雑な文書構造を表現できます．

（1）リスト（UL，OL）

```
<!ELEMENT UL - - (LI)+ >
<!ELEMENT OL - - (LI)+ >
<!ELEMENT LI - o (%インライン系要素; | %ブロック系要素;+ ) >
<!--
        UL や OL は内部に LI しか記述できない.
        LI は UL や OL の直下のみに記述できる.
        LI の終了タグは場合によっては省略できる.

        LI の子要素は，インライン系要素１つかブロック系要素１つ以上.
-->
```

UL（Unordered List）要素は箇条書きリスト，OL（Ordered List）要素は数え上げリストに相当します（図3-3）．

図3-3. リスト

箇条書きリスト（UL）
- りんご
- ごりら
- らっぱ

数え上げリスト（OL）
1. りんご
2. ごりら
3. らっぱ

UL，OLの中には，項目に相当するLI（List Item）要素だけが記述できます．LI要素の中には任意のブロック系要素もインライン系要素も記述できます．したがって，LIの中にさらにリストを作成することも可能です（図3-4）．

LIは，ULあるいはOLの直の子要素としてのみ記述可能です．このことには十分注意してください．

図3-4. リストの入れ子

```
<P>「果物リスト」</P>
<UL>
  <LI>
      <P>みかん</P>
      <OL>
          <LI>三ヶ日みかん</LI>
          <LI>四日市みかん</LI>
      </OL>
  </LI>
  <LI>
      <P>なし</P>
      <OL>
          <LI>二十世紀なし</LI>
          <LI>幸水</LI>
      </OL>
  </LI>
</UL>
```

「果物リスト」
- みかん
    1. 三ヶ日みかん
    2. 四日市みかん

```
・なし
    1. 二十世紀なし
    2. 幸水
```

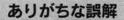
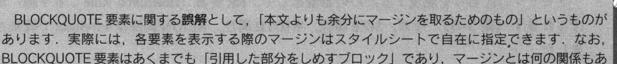
HTML3.2以前では，LIの中に見出しやアドレスを記述できませんでした．HTML4.0からは文法的には可能ですが，互換性のために記述しない方がよいでしょう．

（2）引用ブロック（BLOCKQUOTE）

```
<!ELEMENT BLOCKQUOTE - - (%ブロック系要素;)+ >
```

HTML文書内にほかの書籍などからの引用を記述する場合，BLOCKQUOTE要素として記述します．BLOCKQUOTE要素内には任意のブロック系要素が記述できます．なお，2章で紹介したQ要素は，ブロック系要素（P要素など）内部に引用語句を入れ込む場合に用います．BLOCKQUOTE要素はブロックごと引用する場合に用います．

```
<P><CITE>日本ユニテックSGMLサロン（1995）はじめてのSGML：285pp，技術評論社，東京都
</CITE>の11ページ<Q>1章：SGMLへの誘い</Q>からの引用です．

<BLOCKQUOTE>
    <H1>1章：SGMLへの誘い</H1>
    <P>この章では，読者のみなさんにSGMLのイメージを
    つかんでいただくことを目的とし，
    SGMLが登場した背景，その利用目的を説明します．</P>
</BLOCKQUOTE>
```

ありがちな誤解

BLOCKQUOTE要素に関する誤解として，「本文よりも余分にマージンを取るためのもの」というものがあります．実際には，各要素を表示する際のマージンはスタイルシートで自在に指定できます．なお，BLOCKQUOTE要素はあくまでも「引用した部分をしめすブロック」であり，マージンとは何の関係もありません．

（3）任意ブロック（DIV）

```
<!ELEMENT DIV - - (%ブロック系要素; )+ >
```

DIV（Division）要素は，「特定の意味を持たない」ブロックです．2章で紹介したSPAN要

素のブロック版です．望みの内容を意味するブロック系要素が用意されていない場合は，DIV要素に適切なCLASS属性を設定して，擬似的に表現してください．

```
<DIV CLASS="CHAPTER1">
<H1>私のPC</H1>
    <P>私のPCの宣伝文句は，かの長島茂男さんがにこやかに語っていました．</P>
    <BLOCKQUOTE>
        <P>「サポート体制で選べば，フローラ.」</P>
    </BLOCKQUOTE>
    <P>そう，日立のフローラです．</P>
    <P>94年に購入したため，いまやローエンドなスペックです．</P>
</DIV>
<DIV CLASS="CHAPTER2">
<H1>私のシーケンサ</H1>
    <P>私のシーケンサの宣伝文句は，たしかこんな感じです．</P>
    <BLOCKQUOTE>
        <P>「ビデオテープサイズの音源一体型シーケンサ.」</P>
    </BLOCKQUOTE>
    <P>そう，ヤマハのQY-10です．</P>
    <P>おもちゃみたいな見た目のわりには，しっかりした音が鳴ります．
    僕は97年現在でも活用しています．</P>
</DIV>
```

3.2.2. 特殊なブロック系要素

（1）水平線（HR）

```
<!ELEMENT HR - o (EMPTY)>
```

トピックの区切りを表現するための要素として，HR要素（Horizontal Rule：水平線）が挙げられます（**図3-5**）．なお，SIZE属性，WIDTH属性，ALIGN属性は4.0-Strictで破棄されました．

図3-5. HR要素

```
...
<P>以上をもって，説明を終わります．</P>
<HR>
<H1>おわりに</H1>
...
```

以上をもって，説明を終わります。

おわりに

3.3.3. 単純ブロック系要素

ここでは，その子要素としてインライン系要素のみを許すブロック系要素を紹介します．2章で紹介したH1～H6要素，P要素もここに区分されます．

(1) アドレス (ADDRESS)

```
<!ELEMENT ADDRESS - - (%インライン系要素;)+>
```

連絡先に相当する要素です．通常は，文書の作者への連絡先を記述します．e-mailアドレスを添えるのが一般的でしょう．

```
<HR>
<P>ご意見ご要望および苦情はE-MAILにて</P>
<ADDRESS>
<A HREF="mailto:ks@foo.bar.or.jp">e-mail to : ks!  (ks@foo.bar.or.jp) </A>
</ADDRESS>
<HR>
```

自然言語の感覚では「連絡先」はインライン系要素のような気がしますが，HTMLではブロック系要素に指定されています．W3Cの仕様書によると，文書の「フッター」的に利用することが想定されているようです．

(2) 事前整形ブロック (PRE)

```
<!ELEMENT PRE - - (%インライン系要素;)+  -(IMG | SUB | SUP )>
<!--
        その子要素としてIMG，SUB，SUPを禁止する．
-->
```

プログラムのソースや詩などでは，特殊な位置での改行や余白が要求されます．それら特殊なブロックに相当する要素をすべて用意するわけにはいかないため，HTMLでは「ソースの

空白，タブ，改行をそのまま表示する」ブロックすなわちPRE（Preformatted Text）要素を用意しています．各種の特殊な改行や余白はあらかじめソース中で整形し，PRE要素と指定してください．

```
<PRE CLASS="JAVACODE">
    public boolean action(Event e, Object o){
        if (e.target == add_b){
            ballManager.add();
            return true;
        } else
        if (e.target == delete_b) {
            ballManager.delete();
            return true;
        } else
        return super.action(e, o);
    }
</PRE>
```

PREブロックの中にはインライン系要素が記述可能ですが，ソースの事前整形を崩す可能性のある要素，すなわちIMGやSUPなどいくつかの要素は排除指定されています．

PRE要素はあくまでも「特殊なケース」だけを担当するべきです．通常の段落はP要素で，リストはUL要素で記述するよう心がけてください．

3-3 some more インライン系要素

ここでは，特殊な使われかたをするインライン要素（A，IMG，BR）を紹介します．

```
<!ENTITY % インライン系要素
    " STRONG | Q | CITE | SUB | SUP | SPAN | A | IMG | BR | #PCDATA">
```

便宜上，#PCDATA もインライン系要素に数えます．

3.3.1. アンカー（A）

```
<!ELEMENT A - - (%インライン系要素;)  -(A) >
<!--
    Aの子要素としてAを禁止する．すなわち，Aの中にAは入れられない．
-->
```

属性名	属性値	省略時の値	備考
HREF	URL	実装依存	リンク先の URL
NAME	文字列	実装依存	要素に文書内での“名前”を指定する

A（アンカー：Anchor）要素には大きく2つの使い方があります．ひとつは，ハイパーリンクを実現するためにHREF（ハイパーリファレンス）属性を用いるものです．もうひとつは，文書内に“内部名”を付けるためにNAME属性を用いるものです．

HREF 属性と NAME 属性の両方を同時に指定することも可能です．ちなみに，両方を省略するのも文法上は可能ですが，そうするとアンカーの機能が無くなってしまいます．

（1）ハイパーリンク

WWWの心臓でもあるハイパーリンクは，このA要素で指定します．ハイパーリンク機能をあえて解説すれば，ユーザがA要素を指示（クリックなど）した際にWWWブラウザがそのHREF属性に指定されたURLを自動的に読み込む機能です．

```
「<P>本書は<A HREF"http://www.gihyo.co.jp/">技術評論社</A>から
出版されます．</P>」
```

「here」症候群

暗黙の了解として，ハイパーリンク先の内容は，表示されているアンカー文字列の内容と関連するものになっています．すなわち，「技術評論社」という文字がアンカー文字列であれば，そのリンク先はhttp://www.gihyo.co.jp/や誰かが技術評論社について書いたHTML文書などであるのが普通です．

逆に言えば，A要素を記述するときは，アンカー文字列の内容を読んだだけで**リンク先が想像できるもの**を選ぶべきです．ということは，

```
「技術評論社に関しては<A HREF="http://www.gihyo.co.jp/">こちら</A>」
```

という書き方は望ましくありません．いったい「こちら」って何でしょう？ これは「here症候群」と呼ばれる一種の流行り（?）病です．次のように修正するべきでしょう．

```
「<A HREF="http://www.gihyo.co.jp/">技術評論社に関してはこちら</A>」

「技術評論社に関しては
    <A HREF="http://www.gihyo.co.jp/">http://www.gihyo.co.jp/</A>」

「<A HREF="http://www.gihyo.co.jp/">技術評論社に関して</A>」
```

また，「こちらをクリック」という表現も望ましくありません．WWWを利用する誰もがマウスを用いているとは限りません．ではどのように表現するべきか？ こたえは，「こちら」という不自然な言葉を排除すれば自ずと明らかでしょう．そう，リンク先を意味する単語だけを書けばよいのです．

このコラムはWAIの指針をもとにしています．http://www.w3c.org/WAI/

（2）内部名

たとえば，http://www.foo.com/bar/test.htmlに次のような記述があるとします．

```
<H1><A NAME="intro">1.はじめに</A></H1>
```

この場合，「http://www.foo.com/bar/test.html#intro」はこの部分に相当します．すなわち，HTML文書に内部名NAMEがセットされている場合，その文書を指すURLに#NAMEを追加することによって，相当する部分を直接指示することができます．WWWブラウザはURL#NAMEを指示されると，その部分を表示しようとします．

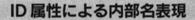
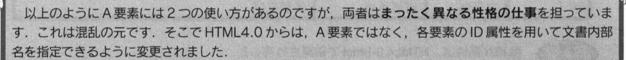
これを利用すれば，リンク付きの目次などを表現できます．

```
<H1>目次</H1>
<UL>
   <LI><A HREF="test.html#intro">1.はじめに</A>
   <LI><A HREF="test.html#main">2.スタイルシートの可能性</A>
   <LI><A HREF="test.html#outro">3.さいごに</A>
</UL>
```

同一文書内で#NAMEへリンクを指定するときは，ファイル名を省略しても構いません．
<A HREF="#intro"> ～ </A>．

内部名を指定する場合でも，A要素の終了タグは省略できません．それにもかかわらず，この終了タグはよく書き忘れられていますのでご注意ください．

ID属性による内部名表現

以上のようにA要素には2つの使い方があるのですが，両者は**まったく異なる性格の仕事**を担っています．これは混乱の元です．そこでHTML4.0からは，A要素ではなく，各要素のID属性を用いて文書内部名を指定できるように変更されました．

```
<H1 ID="intro">1.はじめに</H1>
```

W3Cは「ID属性のほうがNAME属性よりも好ましい」とは明言していませんが，筆者は「ID属性が望ましい」と考えています．これからは，A要素はハイパーリンクにのみ利用しましょう．

ID属性は，CLASS属性と同じくBODY要素のほぼすべての子要素に指定できます．しかし，Netscape Navigator4.0はこの機能をサポートしていません（見当違いのところを表示してしまいます）．Internet Explorer4.0はサポートしています．なお，互換性確保のために，A要素のNAME属性もこれまでどおり活用できます．

アンカーの入れ子

アンカー要素の中にはインライン系要素が記述できますが，A要素は排除されています．すなわち，アンカー同士を入れ子にすることは不正です（**図3-6**）．

図3-6. 不整なA要素

```
不正なアンカー例：
    <P>もし<A HREF="music.html">音楽に興味があれば，
    <A HREF="zappa.html">とくにフランク・ザッパに</A>興味があれば，
    </A>私の書いた文章を読んでください．</P>
```

このルールは，NAMEとして用いる場合でも変わりません．

3.3.2. インライン画像（IMG）

```
<!ELEMENT IMG - o (ENPTY) >
```

属性名	属性値	省略時の値	備考
SRC	URL	省略不可	画像ファイルを指すURL
ALT	文字列	省略不可	代替文字

ALIGN属性は，HTML4.0-Strictで破棄されました．

HTMLのマークアップ自体では絵をかけませんが，既存の画像ファイルを文書に埋め込むことは可能です．画像を指定する要素はIMG（Image）要素です．画像は語句ではありませんので，空要素です．

（1）SRC属性

個々のIMG要素は，実際にはSRC（ソース：source）属性で指示される**画像に置き換えられて**表示されます．「どんな画像形式のファイルでも指定できる」とは言えませんが，W3CはGIF，JPEG，PNGを想定しています．なお，PNGはW3Cが推奨する画像フォーマットです．余談ですが，MOSAICの時代には，インライン画像はGIFのみでした．

PNGとは

PNG（Portable Network Graphics：発音は「ping」と同じ）は，GIFやJPEGなどと同じく，画像圧縮形式のひとつです．フルカラーRGB情報に加えてα値（色の透明度）も保持でき，γ（輝度）補正も可能になっています．GIFと同様にインタレース表示が可能です．また，JPEGとは異なり，完全可逆圧縮になっています．

一般利用者には関係のない話かもしれませんが，じつはGIFフォーマット（の内部のLZW圧縮アルゴリズム）はユニシスおよびコンピュサーブの特許になっており，GIFを扱うソフトウェアを販売する場合には，ロイヤリティを支払わなければならないことになっています．彼らはGIFが世界に浸透した後で，突然ロイヤリティを請求し始めたため，業界では大変な物議になりました．そこで，「GIFよりも優れていてしかも無料のものを作ろう!」と開発されたのがPNGなのです．PNGの名前には，「PNG's Not GIF」という叫びが込められています．

PNGの仕様はW3C Recommendationとして公開されており，誰もロイヤリティを要求できません．ちなみに，Internet Explorer4.00以降もNetscape Navigator4.04以降もPNGを表示できます．なお，Netscape Navigator2.0～3.xも，プラグインによってPNGを表示可能です．

PNGの情報は，以下のサイトから入手できます．各種画像をPNGフォーマットに変換できるソフトウェアも，同サイトで紹介されています．

```
http://www.w3.org/TR/REC-png.html
http://www.w3.org/Graphics/PNG/
http://www.cdrom.com/pub/png/
```

（2）ALT属性

画像を表示できない場合でも何らかの情報を提供するために，IMG要素にはALT（alternative）属性が用意されています．画像を表示しないWWWブラウザは，**画像の代わりに**ALT属性に指定された文字列を表示します．したがって，ALT属性に画像を説明するような文を指定しておくと，より多くの人に情報を提供できることになります．とくに，画像にハイパーリンクを仕掛けてある場合は，この情報が重要です．

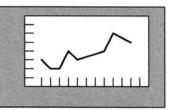
ところで，代替文が必要ない場合は，「**ALT=""**」と記述してください．これは「この画像にはとくに意味はありません」という積極的なアピールです．先に挙げたような画像を処理できないWWWブラウザは，この指定を元に不必要な情報を除去します（**図3-7**）．

```
<H1><IMG SRC="mark.gif" ALT="">業績発表</H1>
<P>今年の業績を，以下のようなグラフとして示します．</P>
<P><IMG SRC="score.gif" ALT="[業績の表]"></P>
<P><A HREF="score.jpeg">より詳しい画像</A>も別途用意しました．</P>
```

図 3-7. IMG 要素の画像 off 時の表現

＜画像on時＞

業績発表

今年の業績を、以下のようなグラフとして示します。

より詳しい画像も別途用意しました。

＜画像off時＞

業績発表

今年の業績を、以下のようなグラフとして示します。

[業績の表]

より詳しい画像も別途用意しました。

余分な画像は要らない

すべての WWW 利用者が画像を表示したいとは限りません．たとえば，画像は文章よりもデータサイズが大きく，ネットワークから取り込むには時間がかかるため，WWW ブラウザを「普段は画像を表示しない」ように設定している人もいます．また，そもそも文字しか表示できない WWW ブラウザや，音声読み上げ式 WWW ブラウザというものも世の中には存在します．このような人たちを考慮して，画像に頼り過ぎない文書を作成することも大事です．

WIDTH 属性と HEIGHT 属性

属性名	属性値	省略時の値	備考
HEIGHT	数値	実装依存	画像表示の縦ポイント数．HTML3.2 から
WIDTH	数値	実装依存	画像表示の横ポイント数．HTML3.2 から

IMG 要素に，HEIGHT 属性/WIDTH 属性を指定すると，画像の表示サイズを変更できます．省略した場合は，WWW ブラウザは「画像本来のサイズ」が指定されていると判断します．

WWW ブラウザは，画像データ読み込み終了よりも前に，この指定の情報をもとにレイアウトを決定し文書を表示します．すなわち，**読者の待ち時間を短くできる**のです．Netscape Nagivator4.0 や Internet Explorer4.0 はこの機能をサポートしています．画像サイズを変更するつもりのない場合でも，サイズを明示しておくとよいでしょう（**図 3-8**）．

図 3-8．HEIGHT 属性，WIDTH 属性のある IMG 要素

業績発表

今年の業績を，以下のようなグラフとして示します。

[業績の表]

より詳しい画像も別途用意しました。

```
<H1><IMG HEIGHT="15" WIDTH="15" SRC="mark.gif" ALT="">業績発表</H1>
<P>今年の業績を，以下のようなグラフとして示します．</P>
<P><IMG HEIGHT="50" WIDTH="30" SRC="score.gif" ALT="[業績の表]"></P>
<P><A HREF="score.jpeg">より詳しい画像</A>も別途用意しました．</P>
```

…といいたいのですが，これらの属性は HTML4.0-Strict では**破棄**されました．その指定はスタイルシートで行うべきだというのです．では，スタイルシートに対応していない WWW ブラウザを用いている人のためにこれらの属性を使いたい場合はどうすればよいのでしょう？　答えは，「過渡期の間は HTML4.0-Transitional を使わざるをえない」です．

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

筆者自身は「そもそも，画像に頼ったレイアウトのある HTML 文書を書いては**いけない**」と考えています．筆者の WWW リソースでは，画像をきちんと見てほしいときは外置き画像（アンカーのリンク先として画像ファイルを指定）にしており，インライン画像は簡単なアイキャッチ用のものばかりです．したがって，ほとんどの IMG 要素は ALT=""であり，あえて WIDTH，HEIGHT 属性は指定していません．「レイアウト情報が重要でないならば，WIDTH や HEIGHT は付けなくてもよいんじゃないかなあ ^^;」というのが筆者の考えです．

3.3.3. 強制改行（BR）

```
<!ELEMENT BR - o (EMPTY) >
```

通常の文は，表示領域の幅にあわせて自動折り返しされます．しかし，何かの事情で強制改行したい場合は，BR（Line Break）要素を利用します．BR 要素は空要素です．

```
<P>古典の<CITE>「平家物語」</CITE>ですっけ，私の記憶が確かならば…</P>
<BLOCKQUOTE>
    <P>
    祇園精舎の鐘の声，<BR>
    諸行無常の響きあり。<BR>
    沙羅双樹の花の色，<BR>
    盛者必衰の理をあらはす。
    </P>
</BLOCKQUOTE>
```

改行は段落ではない

BR 要素はあくまでも強制改行であって，段落を作る要素ではありません．

そもそも，段落は「改行」で作られる，というのは単なる思い込みです．たとえば，新聞の小さなコラム欄のように，段落で改行せずに「▼」マークを示すだけの場合もあります．

HTML 文章中の段落と「P 要素」で示しておけば，その表示方法はスタイルシートで自由に変更ができます．この「新聞コラム」も表現可能です．

日本語の標準的な段落：

```
P{
    display : Block;
    margin-bottom : 0em;
    text-indent : 1em;
}
```

新聞コラムなどの段落：

```
P{
    display : inline;
}
P : before {
    content : "▼" ;
}
```

一方，BR 要素で書いてしまっては，このような調整は不可能です．HTML 本来の思想に基づいて P と BR を正しく使い分けてください．

HEAD要素の子要素

HEAD要素の子要素は，2章で紹介したTITLE要素以外はすべて応用的な要素です．その機能はあまりに多岐にわたるために，本書の範疇を超えます．そこで本書では，説明をスタイルシートを用いる上で必要な範囲に限定します．とはいえ，本章では文書型定義的な紹介にとどめ，活用のための解説は4章で行います．

```
<!ELEMENT HEAD o o ( TITLE )   +( STYLE | LINK ) >
<!--
    STYLE,LINKは順序を問わず何個でも記述できます.
-->
```

3.4.1. STYLE要素

```
<!ELEMENT STYLE - - (%スタイルシート;)>
```

属性名	属性値	省略時の値	備考
TYPE	文字列	省略不可	スタイルシートの種類を明示．4.0より
MEDIA	文字列	実装依存	再生メディアの指定
TITLE	文字列	実装依存	文書内におけるスタイルシートのタイトル

スタイルシートをHTML文書内部に記述するときに用いる要素です．TYPE属性，MEDIA属性，TITLE属性の意味を含め，詳しくは「4.3.HTML文書との連携」を参照してください．

3.4.2. LINK要素

```
<!ELEMENT LINK - o (EMPTY) >
```

属性名	属性値	省略時の値	備考
HREF	URL	実装依存	リンク先URL．省略しては意味をなさない
REL	文字列	実装依存	現在の文書から見たリンク先の説明
REV	文字列	実装依存	リンク先から見た現在の文書の説明
TYPE	文字列	実装依存	データの種類を明示．4.0より
MEDIA	文字列	実装依存	再生メディアの指定．4.0より
TITLE	文字列	実装依存	リンク先を指す名前

LINK要素は，現在の文書と関連するリソースを列挙するための要素です．どのような意味で関連しているのかは，LINK要素のREL（Forward Relationship），REV（Reverse Link）属性で指定します（**表3-2**）．スタイルシートファイルも「関連文書」の一種としてLINK要素で指定できます．詳しくは「4.3. HTML文書との連携」で解説します．

表3-2．HTML4.0仕様書が紹介するREL，REV属性の属性値

キーワード	意味
ALTERNATE	現在の文書の代替文書一別言語訳など
STYLESHEET	スタイルシート
ALTERNATE STYLESHEET	代替スタイルシート
START	現在の文書が所属する一連の文書の「はじめ」
NEXT	次の文書
PREV	前の文書
CONTENTS	現在の文書が所属する一連の文書の目次
INDEX	現在の文書の目次
GLOSSARY	現在の文章に対する「用語集」
COPYRIGHT	現在の文書の版権を説明する文書
CHAPTER	現在の文章の中の「章」になる文書
SECTION	現在の文章の中の「節」になる文書
SUBSECTION	現在の文章の中の「項」になる文書
APPENDIX	現在の文書が所属する一連の文書の「索引」
HELP	現在の文書に関するヘルプ文書
BOOKMARK	ブックマーク （いわいるリンク集で，文書に対する「索引」のようなもの）

スタイルシートに限らず，NEXT や INDEX を適切に記述しておくと，文書ナビゲートに関していくつかの利点が得られます.

・印刷時に WWW ブラウザが関連文書を列挙できる
・ユーザが現文書を読んでいる間に，WWW ブラウザは関連文書ファイルを先読みできる

```
<HEAD>
<TITLE>TEST</TITLE>
<LINK REL="STYLESHEET" TYPE="text/css" MEDIA="SCREEN" TITLE="STANDARD"
      HREF="normal.css">
<LINK REL="NEXT" TITLE="SECTION2" HREF="foo2.html">
</HEAD>
```

CSS syntax in detail

2・3章では，HTML文書からスタイル情報を締め出すことで，単純かつ一般性のある文書になることを解説しました．今後の章では，スタイル情報をスタイルシートとして構築する手順を解説します．

4章では，スタイルシート記述の文法的な側面を解説します．ただし，文法の全てを知っていなければスタイルシートを書けないか，といえばそうでもありません．先に6章「road to mastering CSS」を読み，関連する文法だけを抜き読みしてもかまいません．

なお，CSS2で新しく採用された仕様には「CSS2」である旨を明示します．現在のところ，WWWブラウザはCSS2をサポートしていないことにご注意ください．

4-1 記述のルール

4.1.1. 文字の扱い

CSS仕様のスタイルシートの記述は，たいへんに「定型的」です．すなわち，URLとフォントファミリーの指定を除き，あらかじめ規定されたキーワードを選択したり，数値を記入するだけでシートは完成します．

シートの一般様式は「要素名 {プロパティ名：指定する値}」です．具体的なプロパティやその値に関しては5章で詳しく解説することにし，4章ではシートそのものを記述するための文法に関して解説します．

まず，スタイルシート記述に用いる文字に関する取り決めは次のとおりです．

・全角半角	半角のみを用いる（コメントは別）
・大文字小文字	区別されない（URLを指定する部分などは別）
・単語の区切り	スペース，タグ，改行（区切りを連続して複数記述しても，1つの区切りとして扱われる）
・コメント	「/*」から「*/」で囲んだ部分（コメントは入れ子にできません．/* test /*test*/ test*/とは書けません）

4.1.2. スタイル指定の一般形式

専門用語として，「スタイル指定が適用される先」を「セレクタ（Selector）」とよび，「指定するスタイル内容」を「宣言（Declaration）」と呼びます．すると，シートの一般様式は，

「セレクタ {宣言}」

になります．

端的には，セレクタはHTMLの要素そのものです．しかし，実際にはいくらかのバリエーションがあります．そのバリエーションを定めるのが「文法」です．まず，**図4-1**に正規の文法をいくらか簡略化したものを提示し，以降で各々のバリエーションについて解説していきます．

図 4-1. CSS の基本文法

記述の説明：

< >　書くべきものの説明
( )　省略可能な語句
( )…　0回以上の繰り返し
|　or を表わす記号：どちらか一方のみを記述
/* */　コメント
それ以外　そのまま書く（ピリオド，コロンなど）

スタイル指定の一般式：

<セレクタ>（, <セレクタ>）…　{<宣言>（；<宣言>）…}
/* セレクタはカンマ（,）で区切ることで，
　宣言はセミコロン（；）で区切ることで，
　グルーピングか可能．
*/

宣言：

<プロパティ値>：<値>　(!important)
|　/* 空の宣言も許されている */
/*!important キーワードは，カスケーディング処理で使われる */

4.1.3. セレクタや宣言のグループ化

同じセレクタに対する宣言をまとめて記述する際には，宣言をセミコロン (;) でつないでグループ化します．また，同じ宣言を複数のセレクタにまとめて記述するには，セレクタをカンマ (,) でつないでグループ化します．

なお，分けて記述してもまとめて記述しても，意味は変わりません．

分けて記述した例：

```
P{font-weight: 500}/* 太さ 500*/
P{text-align: left}
CITE{color: red}
STRONG{color: red}
CITE{font-style: italic}
```

まとめて記述した例：

```
P{
    font-weight: 500;
    text-align: left;
}  /* 最後の宣言にはセミコロンは不要ですが，
    付けても不正ではありません．*/
CITE, STRONG{color: red}
CITE{font-style: italic}
```

4.1.4. 宣言が衝突した場合の処理

宣言内容が衝突した部分—同じ要素，同じプロパティに複数の指定がなされた場合—は，後になされた宣言のみが有効になります．

受け取ったシート：

```
P{
    text-align: center;
    color: black;
}
P{
    text-align: left;   /* 同じ要素・同じプロパティの宣言が既にある */
}
```

有効になる宣言：

```
P{
    color: black;
}
P{
    text-align: left;   /* もっとも後ろのものが採用される */
}
```

ただし，!important キーワードが付けられていた場合，優先順位は変動します．

受け取ったシート：

```
P{
    text-align: center !important;
    color: black;
}
P{
    text-align: left;   /* 同じ要素・同じプロパティの宣言が既にある */
}
```

有効になる宣言：

```
P{
    color: black;
}
P{
    text-align: center;   /*!importantのうち，もっとも後ろのものが採用される*/
}
```

 importantキーワードは，後述のカスケーディング処理でも用いられます．

4.1.5. 属性値による対象の限定

(1) CLASSとID

CSSでは，要素全般への指定だけでなく，HTML文書中でのIDやCLASS属性を特定したスタイル指定が可能です．IDはシャープ（#）で，クラス名はピリオド（.）で指定します（CSS2では，「#が取り扱うのはHTMLのID属性ではなく，属性値がID機能を持つすべての属性である」と拡張されています．この拡張は，XMLを考慮に入れたものでしょう）．

一般例：

```
<要素名>.<クラス名>  {<宣言>}
<要素名>#<ID>  {<宣言>}
```

宣言例：

```
BLOCKQUOTE{  /*クラス名を問わずに適用されるスタイル*/
    font-style:italic;
    margin: 0.2em 2em;/*上下マージン0.2em，左右マージン2em*/
}
BLOCKQUOTE.BEATLES{color: red}  /*クラスBEATLESのみの追加スタイル*/
BLOCKQUOTE.STONES{color: blue}
```

 emは「一文字分」を意味しています．

(2) その他の属性による限定

CSS2では，CLASS，ID属性に限らず，すべての属性をもちいて対象セレクタを限定できるようになります．

▶「要素名[属性名]」…その属性が宣言されている場合に有効．値は問わない

```
A[HREF]{font-style: italic}…リンク先を指定したアンカー
A[NAME]{font-weight: bolder}…内部名を指定したアンカー
```

▶「要素名[属性名＝属性値]」…その属性値が指定されている場合にのみ有効

```
H1[ALIGN="CENTER"]{text-align:center;}
A[HREF="index.html"]{color: red;}
A[rel="NEXT"]{color: blue;}
```

▶「要素名[属性名˜＝属性値]」…属性値がスペース区切りの値リストの場合に，その値の中に1つでも指定した属性値のものがあれば有効（主にCLASS属性）

```
「P[class~=LOVE]{color:red}」ならば，
    ○…<P CLASS="LOVE">
    ○…<P CLASS="LOVE PEACE SPECIAL">
```

▶「要素名[属性名|=値]」…属性値がハイフン区切りの値リストの場合に，その値の中に一つでも指定した属性値のものがあれば有効（主にLANG属性で，言語別に指定を切り分ける場合などに使われる．「:lang」疑似クラスを参照）

```
    BLOCKQUOTE[lang|=fr] { quotes: '≪ ' ' ≫' }
    BLOCKQUOTE[lang|=ja] { quotes: '「' '」'}
```

（3）クラス名による限定の拡張

CSS2では，1つの要素に複数のクラスが指定されていることを考慮し，クラス名へのアクセスを強化しました．

▶「要素名.クラス名1.クラス名2」…クラス名を複数指定した場合，class属性値の並びの中に指定したすべてのクラス名を含んでいる場合のみ有効

```
「P.LOVE.PEACE{color:red}」ならば，
    ×…<P CLASS="LOVE">
    ○…<P CLASS="LOVE PEACE SPECIAL">
```

（4）要素名を指定しない宣言

CSS1では，IDやクラス名だけを指定した宣言（要素名を省略した宣言）を行ったならば，すべての要素に関する指定を行ったことになります．

要素を問わない指定：

```
.<クラス名> {<宣言>}
#<ID> {<宣言>}
```

例：

```
/*BLOCKQUOTE でも P でも，CLASS="BEATLES" ならば色を変える */
    .BEATLES{color: red}
/* 要素は問わずに ID="CHAPTER2" に指定 */
    #CHAPTER2{color: blue;}
```

CSS2 では，要素名を「*」と書いた場合に，「すべての要素に対する宣言をした」と解釈することに変更されました．なお，上位互換のため，CSS1 と同じ記述方法も可能です．CSS2 における「*」は過度に強力な仕組みで，4.2 で説明する「継承」の利点を無視した指定すら可能になります．使用する際には，本当に必要なのかどうかを十分検討してください．

例：

```
/* 強制的に，すべての要素の色を指定 */
    * {color: balack; background: white}
```

4.1.6. CSS1 における文脈判断

普通のスタイル宣言は，指定した要素が文書中のどこに現れてもすべてに適用されます．すなわち，

```
    STRONG{ color: red;}
```

とさえ記述すれば，STRONG 要素が H1 中に現われようと P 中に現われようと，文脈を問わずに赤色になります．

しかし，「見出しの中の STRONG 要素だけ色を変えたい」ケースもあるでしょう．たとえば，H1 要素も「color: red」と指定されているため，同じ STRONG でも「H1 に含まれる場合だけ」青色にしたいというケースが考えられます（**図 4-2**）．

CSS1 では，「対象要素が特定の親要素に含まれた場合にのみ有効になるスタイル」を，「親要素」「子要素」の順に「スペース」で区切って並べたセレクタを用いて指定できます．これを**文脈付きセレクタ（Contextual Selector）**と呼びます．

一般形式：

親要素 子要素 （ 孫要素）…{<宣言>}

宣言例：

```
/* 単純宣言 */
    BODY{
        color: black;
        font-size: 10pt;
    }
    H1{
        color: red;
        font-size: bolder;
        text-decoration: underline;
    }
    STRONG{   /* すべての STRONG に適用されるスタイル */
        color: red;
        font-style: italic;
    }
/*H1 に含まれる STRONG にだけに対する“追加の”スタイル */
    H1 STRONG{color: blue;}
```

図 4-2. H1 に含まれる STRONG だけへの追加指定

```
<H1> 私の <STRONG> 趣味 </STRONG> について </H1>
<P> 私の趣味は <STRONG> 音楽 </STRONG> です. </P>
```

私の***趣味***について

私の*趣味*は音楽です.

4.1.7. CSS2における文脈判断

CSS2では，文脈判断がCSS1よりも数段強力になりました．

（1）子孫セレクタ（下降セレクタ，descendant selector）

これは，CSS1における文脈付きセレクタと全く同じものです．さきほどは明言しませんでしたが，子孫セレクタにおける「親子関係」は，直接の「子」には限定されません．たとえば，「BLOCKQUOTE STRONG」とは，BLOCKQUOTE要素に含まれるすべてのSTRONG要素に対する指定であり，次のSTRONG要素にも適用されます．

```
<BLOCKQUOTE>
    <P>私の趣味は<STRONG>音楽</STRONG>です．</P>
</BLOCKQUOTE>
```

文脈指定がぶつかった場合

もし，「BLOCKQUOTE STRONG」と「P STRONG」の両方にスタイルを宣言していれば，上記の文脈では両方が同時に有効になります．ただし，衝突する宣言に関しては，より内側の文脈 ― この場合は「P STRONG」 ― に対する宣言が優先されます．

存在する指定：

```
    BLOCKQUOTE STRONG{
        color: black;
        font-weight: bold;
    }
    P STRONG{
        color: red;
    }
```

実際に「BLOCKQUOTE P STRONG」に適用される指定：

```
    {
        font-weight: bold;
        color: red;
    }
```

（2）子セレクタ（child selector）

子孫セレクタとは異なり，ある特定の要素が直接の親要素になった場合にだけ有効になる宣言を行いたい場合は，この「子セレクタ」を用います．

一般形式：

```
親要素 > 子要素 {<宣言>}
```

宣言例：

```
DIV.BEATLES > P{font-family: sans-serif;}
```

例ソース：

```
<P>もちろん，僕だって BEATLES は嫌いじゃない．(←ここには適用されない)
<DIV CLASS="BEATLES">
    <P>好きなのは，初期のロックもの．ハード・デイズ・ナイトなどには感涙．
    しかし，次にリストアップする曲は嫌い．(←ここには適用される)
    <UL>
        <LI><P>イエスタデイ (←ここには適用されない)
        <LI><P>ヘイ・デュード (←ここには適用されない)
        <LI><P>ロング・アンド・ワインディング・ロード (←ここには適用されない)
    </UL>
    <P>中期以降でも，やはりサージェント・ペパーズなどは好き．(←ここには適用される)
</DIV>
```

(3) 隣接セレクタ (Adjacent sibling selector)

親子関係ではなく，文書中の連続関係に注目したのが「隣接セレクタ」です．たとえば，「H1 要素直後の H2 要素」などに指定したい場合に用います（一般に，見出しが上位の見出しに連続する場合と地の文章に続く場合とで，マージンなどを変えることは多いでしょう）．

一般形式：

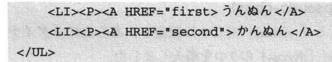
```
直前要素 + 直後要素 {<宣言>}
```

宣言例：

```
H1 + H2{margin-top:0em;}
H2{margin-top:1em;}
```

例ソース：

```
<H1>すみけんリソース!</H1>
<H2>前書き</H2><!--←ここは H1+H2-->
<P>うんぬんかんぬん

<H2>目次</H2><!--←ここは H1+H2-->
<UL>
```

```
    <LI><P><A HREF="first> うんぬん </A>
    <LI><P><A HREF="second"> かんぬん </A>
</UL>
```

(4) 複雑な指定の場合の優先順序

各種文脈の指定は，複数の種類を混ぜることが可能です．その場合でも，指定は単純に前から順に読み解くだけであり，特別な優先順位は存在しません．

・「A B.NAME > C D{宣言}」…要素Dで要素Cの内部のもの，ただし要素Cは要素Bの直下の子要素であること，ただし要素Bは要素Aの内部のもので，クラス名として“NAME”が指定されていること．

4.1.8. 疑似クラス

(1) 疑似クラス (pseudo-class) とは

WWWにとって有用な情報が，必ずしもHTML文書インスタンスそのものから判断できるとは限りません．たとえば，アンカーには「まだ表示したことの無いリンク」「既に表示したことのあるリンク」などの性質の違いがあり，その性質に応じてスタイルを変えることは合理的ですが，その違いはWWWブラウザの履歴情報によってはじめてわかるものです．したがって，いくらHTML文書を眺めても判断できません．

このような場合に活用されるのが「疑似 (pseudo) クラス」です．「疑似クラス」は，文書中には実在しない性質に着目してスタイルを宣言するための手段です．

(2) リンク疑似クラス (link pseudo-class)

CSS1では，上述した「アンカーに対する3つの疑似クラス」のみが用意されています．

アンカーの疑似クラス：

疑似クラス名	説明
:link	まだ表示したことのないURLを指すリンク
:visited	既に表示したことのあるURLを指すリンク
:active	ユーザが選択中のリンク（マウスで触っているなど）

疑似クラスへのスタイル宣言は，要素名と疑似クラス名をコロン（：）でつないで記述します．

```
A:link{color: bule}
A:visited{color:lime}
A:active{color:red; background: #e88;}
```

(3) ダイナミック疑似クラス (dynamic pseudo-class)

CSS2では，リンク疑似クラスを「:link」と「:visited」の二つに限定し，あらたに「ダイナミック疑似クラス」として「:active」「:hover」「:forcus」の3つを導入しました.

疑似クラス名	説明
:active	現在選択中の要素（マウスで触っている，スペースキーを押している，など）
:hover	マウスなどがその上に来ている要素
:forcus	キーボードなどでフォーカスが与えられた要素

疑似クラスを複数用いることによって，「リンク済みでアクティブ」「未リンクでアクティブ」を表現できます.

```
A:link:active{color:white; background: #555}
A:visited:active{color:red; background: #955}
```

「:focus」は，HTML4.0で強化されたアクセスキー（キーボードショートカット）設定に対応するもので，フォームやリンクで活用されるでしょう．また，outlineプロパティと「:forcus」疑似クラスを組み合わせることで，イメージマップ中の選択された部分に枠をつけるなどの表現も可能になります.

なお，リンク疑似要素はアンカー専用の疑似要素でしたが，ダイナミック疑似要素はすべての要素に適用されます．たとえば，「マウスをかざした部分だけ色が変わる目次」などに応用が可能です.

```
UL.CONTENTS LI:hover{color:black; background:#f8f;}
```

(4) 言語疑似クラス (:lang() pseudo-class)

CSS2で追加された疑似クラスで，言語属性を表現するものです．引用符記号など，言語に依存する表現を使い分けたいときに有用です（文書の言語属性は，対象にする要素に直接LANG属性が指定していなくても，META属性や親子関係から判断できます．この意味で，[lang|=us]とは適用範囲が異なります).

```
:lang(fr) > Q { quotes: '≪ ' ' ≫' }
:lang(ja) > Q { quotes: '「' '」'}
```

(5)「最初の子」疑似クラス (:first-child pseudo-class)

CSS2で追加された疑似クラスで，同列の子要素のうち，文書中の順序が一番始めである要素を表現するものです.

例：

```
P > STRONG:first-child{ font-size: larger}

<P>いいですか，僕が <STRONG>「オイッス」</STRONG> といったら，
みんなも「オイッス」と応えるのが礼儀です．
また，儀式として，僕は <STRONG>「声が小さい，もう一度！」</STRONG> と言い返すので，
ここで皆さん笑ってください．
```

4.1.9. 疑似要素

（1）疑似要素とは

疑似要素も，疑似クラスと同様に，実際には文書中に存在しない要素を操作するものです．あえて性格を要約すれば，「書こうと思えば文書中に記述できるが，規則性を持って自動処理することが望ましいもの」が疑似要素に設定されています．

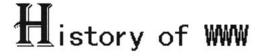
疑似要素へのスタイル宣言も，疑似クラスと同じく，要素名と疑似クラス名をコロン（：）でつないで記述します．

（2）「はじめの文字（:first-letter）」「はじめの１行（:first-line）」

CSS1では，「はじめの文字（:first-letter）」「はじめの1行（:first-line）」の2つの疑似要素が用意されています．その名のとおり，「はじめの文字」「はじめの1行」だけを異なる見栄えで表示したい場合に利用します（**図4-3**）．

図4-3．first-letter疑似要素の使用例

```
H1:first-letter{font:2em fantasy}
```

“<H1>History of WWW</H1>” の表示イメージ：

（3）「前（:before）」「後（:after）」

CSS2で追加された疑似要素で，各要素の前後に，自動的に文字などを挿入したい場合に用います．各疑似要素で表示したい内容は，専用のプロパティ「content」で指定します（7章を参照）．

```
H1:before{content: "一章一"; font-size: 1.5em: display:block;}
BLOCKQUOTE:before{content: open-quote}
BLOCKQUOTE:after{content: close-quote}
BODY:after{content:"以上、終了. "; margin-top: 5em;}
```

4.1.10. @media ブロック

たとえば，一般的な画面（screen）と異なり，文字端末（tty）では文字色，サイズを一定にして表現しなければなりません．このような違いを把握したうえで，メディアごとに指定値を切り替える必要があります．CSS2では，メディアを指定してシートを記述するために，@mediaブロックが用意されています．

CSS1には@mediaに相当する仕組みはありませんが，シートの想定メディアはHTMLのLINK要素で指定すればよいため，とくに困りません．互換性確保のこともあり，今のところは1つのシートは特定のメディアのみを想定して記述しておき，HTML文書からメディアを指定するほうが得策でしょう．なお，具体的なメディアの特徴およびLINK要素に関しては4.3.5.を参照してください．

```
@media screen{
    H1{ font-size: 3em}
    H2{ font-size: 2em}
}
@media screen, tty{
    H1{
        text-align: right;
        margin-top: 2em;
    }
    H2{
        text-align: center;
        margin-top: 1em;
    }
}
```

4.1.11. @import 宣言

スタイルシートは，基本的にはテキストファイルとして作成・保存されています．各々のシート同時は独立した存在ですが，あるシートに別のシートを取り込むことも可能です．

拡張子「.css」

CSS によるスタイルシートファイルの拡張子は「.css」としてください．これはコンテントタイプ（MIME タイプ）指定 “text/css” とからんだ取り決めです．くわしくは，RFC2318 を参照してください．

@import キーワードは，既存のスタイルシートファイルの取り込みを宣言します（C 言語の #include に似ています）．WWW ブラウザは，受け取ったスタイルシートファイルに@import が書かれていれば，そのシートも同時に読み込みます．

@import の指定法：

```
@import "スタイルシートを指すURL";
@import url(スタイルシートを指すURL);
```

@import で取り込んだ後に独自の宣言を書き込むことで，指定を追加，上書きできます．

```
@import url(http://www.w3.org/pub/WWW/def-style.css);
@import url(my-style.css);
STRONG{color: red}  /* このシート独自の追加宣言 */
```

@import を複数指定することも可能ですが，シート独自の追加宣言よりも前に記述する必要があります．なお，@import で取り込んだシート間で宣言が衝突した場合は，単純にもっとも後ろに書かれた宣言のみが有効になります（4.1.4.を参照）．

4-2 継承とカスケーディング

4.2.1. 継承（inheritance）の利点

HTML文書インスタンスには多くの要素が書かれています．また，スタイルシートで調整できるプロパティも多岐にわたります．では，すべての要素にすべてのプロパティについての指定を行わなければならないのでしょうか？

指定の軽量化と合理化のために，CSSは「継承」という仕組みを提唱しています．継承とは，HTML文書インスタンスの構成に沿って，親要素から子要素へスタイルを伝播させる仕組みです．

- 各要素は，まず文脈上の親要素に対するスタイル宣言を「継承」する．
- 継承した上で，自分への宣言を「追加・上書き」する．

具体例を用いて説明しましょう（**図4-4，4-5**）．

図4-4．例文書の文書構造

```
<BODY>
<H1>日記<STRONG>1997年11月24日</STRONG></H1>
<P>TVで偶然に<STRONG>西行法師</STRONG>の短歌を聞いた．</P>
<BLOCKQUOTE>
    <P>「ねがわくば，桜の元で，春死なん」</P>
</BLOCKQUOTE>
<P>本当はこの後に下の句が続くのだが，忘れてしまった．
しかし，これだけでも大変見事に「日本人の風情」を表現していると思う．</P>
</BODY>
```

```
BODY{
    color: black;
    font-size:10pt;
}
H1{
    font-size:15pt;
```

```
    font-family: sans-serif;  /*ゴシック系を期待*/
}
BLOCKQUOTE{    font-style: italic}
STRONG{    font-weight: bold}
```

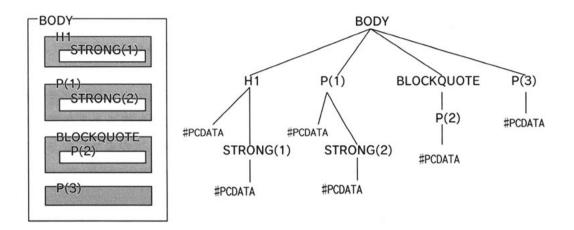



図 4-5. スタイル適用例

日記 **1997年11月24日**

TVで偶然に**西行法師**の短歌を聞いた。

「ねがわくば，桜の元で，春死なん」

本当はこの後に下の句が続くのだが，忘れてしまった。しかし，これだけでも大変見事に「日本人の風情」を表現していると思う。

この文脈におけるH1は，まずBODYの宣言を継承し，

```
    color: black;
    font-size: 10pt;
```

に設定されます．その後，自分への宣言を追加・上書きします．したがって，結果として，

```
    color: black;
    font-size: 15pt;
    font-family: sans-serif;
```

を採用します．

図4-5におけるH1に関するスタイル宣言は，「フォントサイズとファミリーの変更は必要だが，色に関してはこだわらない」という考えで記述されています．すると，色は「継承」によって親要素のものが自動的に適用されるのです．すなわち，シート記述者は，必要な要素に対して必要なプロパティのみを宣言するだけでよいのです．

▶ その要素でとくに変更したいプロパティのみを宣言すればよい

「継承」の，この利点により，スタイルシートは簡潔でわかりやすいものになるでしょう．

次に，STRONG要素を見てみましょう．一口にSTRONG要素といっても，「H1に含まれているSTRONG(1)」と「Pに含まれているSTRONG(2)」とでは，それぞれ継承する親要素が異なるため，最終的に採用するスタイルが異なります．明示するならば，次のようになります．

```
STRONG(1){
    color: black;  /*H1…BODY*/
    font-size: 15pt;  /*H1*/
    font-family: sans-serif;  /*H1*/
    font-weight: bold;
}
STRONG(2){
    color: black;  /*P…BODY*/
    font-size: 10pt;  /*P…BODY*/
    font-weight: bold;
}
```

各要素の親要素は固定ではなく，HTML文書の文脈によって変ります．このすべてのケースを明示してスタイルを調整していくのは大変です．しかし，「継承」によってこれらは自動的に調整されるのです．これも「継承」の利点としてあげてよいでしょう．

▶ 指定しなかったプロパティは，文脈に合わせて適切に調整される

もし「継承」の仕組みが存在しなければ，必要となるすべての文脈に関してあらかじめスタイルを宣言するはめに陥ります．しかも，文書が異なれば必要な文脈は変わってしまうため，「特定のHTML文書にしか利用できない」スタイルシートが出来上がるでしょう．このように考えてみると，「継承」はシートの汎用性/再利用性の確保にも役立っています．

さらに，「すべての文脈について，すべてのプロパティを宣言している」場合に，H1要素のフォントサイズを変更したくなったとしたら，関連していくつの宣言を修正することになるでしょう? この例では「H1 STRONG」だけですが，文書が異なれば，さらに増える可能性があり，修正はだんだんと面倒なものになるでしょう．このように考えると，「継承」はメンテ

ナンスしやすさ－メンテナンス性－の向上にも役立っています．

シート記述者は，「継承」という単純なルールが産み出すさまざまな利点を十分意識する必要があるでしょう．

宣言記述の順

継承はあくまでもHTML文書インスタンスの入れ子構造にしたがって行われるので，宣言記述の順番を入れ替えても継承処理には何の影響もありません．

```
P{
    color: white;
}
BODY{
    font-size: 12pt;
    color: red;
}
```

この場合でも，P要素は先にBODY要素から指定を受け継ぎ（BODYは本文の最上位の親要素ですから，常にその指定は継承されます），その後，P要素に直接指定された宣言を受け入れます．したがって，Pのcolorプロパティは上書きされます．

しかし，「後に書かれた方が有効になる」ように注意して書いたほうがわかりやすいのではないでしょうか．

```
BODY{
    font-size: 12pt;
    color: red;
}
P{
    color: white;
}
```

このように書かれていた方が，「PはBODYの子要素であり，BODYの指定を継承する．継承した上で，colorは上書きされる」という関係が理解しやすいと思われます．筆者は，文書型定義における親要素から順に宣言を記述することを奨励します．

4.2.2. 継承されないプロパティ

継承は大変便利な仕組みですが，すべてのプロパティが継承されるわけではありません．なぜなら，継承するべきでないプロパティも存在するからです（継承されるのか否かは，プロパティごとに仕様書で定められています）．

たとえば，背景画像の指定は継承されません．仮にBODYの背景画像がPに継承されたとすると，Pが現われるたびに背景が書き直され，**図4-6**に示すような結果になります．どちらかといえば，「背景画像の付け直し」よりも「すでに表示されている背景の上に文字を表示する」方が，すなわち，背景は継承されないほうが標準的だと考えられます（CSSのbackground-colorプロパティの初期値はtransparent（透明）になっており，親要素の背景をそのまま表示することが想定されています）．

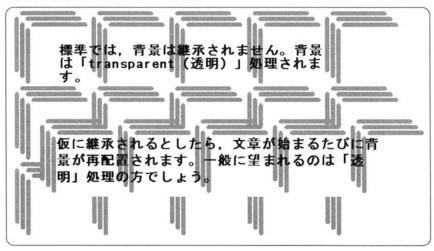
図4-6．背景が継承されるとしたら

また，“直前のテキスト”との垂直位置関係（上付き下付き）も継承されません．たとえば，**図4-7**のHTMLコードのSUPの上付き効果をSTRONGが継承してしまえば，結果は「上付きの上付き」すなわち「eの（2のx乗）乗」のように表示されてしまいます．

図4-7．eの（2のx乗）乗？

```
e<SUP>2<STRONG>x</STRONG></SUP>
```

$$e^{2x} \cdots e^{2^{x}}$$

？？

Tips CSS2では，すべてのプロパティの属性値として「inherit」が用意されました．これを明示した場合は，「通常では継承されないプロパティ」であっても値が継承されます．

4.2.3. カスケーディング（cascading）とは

CSSの仕様は，HTML文書に対して「WWWブラウザ標準のスタイルシート」「ユーザ（読者）が標準とするスタイルシート」「HTML文書作成者が指定するスタイルシート」の3つの

スタイルシートが用意され，この3者の指定を融合させることを想定しています．この3種類のスタイルシートの宣言の融合法，指定が衝突したときの混乱解消法を“カスケーディング”と呼びます．

CSSが想定するカスケーディング処理の基本は，単純で明快です．

CSSでは，各スタイルシートの指定に優先順位をつけています．それを数学の大小記号であらわすと，「WWWブラウザ」<「ユーザ（読者）」<「HTML文書作成者」です．それぞれのスタイルシート間で衝突するプロパティ値が宣言されているときは，優先順位が高い方の指定だけを採用します．宣言が衝突しない場合は，単にそれを採用します．これが，カスケーディング処理の基本です（すべてのカスケーディング処理が終わった後，スタイルの継承が処理されます．それでも指定されていないプロパティがあれば，CSS仕様に定められる初期値が採用されます）．

▶ 優先順位…「WWWブラウザ」<「ユーザ（読者）」<「HTML文書作成者」
▶ 宣言が衝突した場合…優先順位が高い方の宣言を採用
▶ 宣言が衝突しない場合…単純にそれを採用

この処理には以下の利点が挙げられます．

・読者もスタイル調整に参加できる
・文書作成者が指定しなかったプロパティに関しては，読者が普段慣れ親しんでいるスタイルに併せることができる

カスケーディング処理の例：

WWWブラウザ標準：

```
BODY{
    line-height: 1em;
    color: black;
}
```

ユーザ（読者）標準：

```
BODY{
    line-height: 1.3em;
    color: white;
    background: black;
}
```

HTML 文書作成者指定：

```
BODY{line-height: 1.5em}
```

採用される値：

```
BODY{
    line-height: 1.5em;  /*作成者の値*/
    color: white;  /*ユーザ（読者）の値*/
    background: black;  /*ユーザ（読者）の値*/
} /*そのほかのものは，初期値が採用される*/
```

ただし，スタイル宣言に「!important」キーワードをつけた宣言に関しては，優先順位が変ります．「HTML 文書作成者」＜「!important 付き HTML 作成者」＜「!important 付きユーザ（読者)」となります（これはCSS2の仕様です．CSS1では，「HTML 文書作成者」＜「!important 付きユーザ（読者)」＜「!important 付き HTML 作成者」となっていました).

WWW ブラウザ標準：

```
BODY{line-height: 1em}
```

ユーザ（読者）標準：

```
BODY{
   line-height: 1.3em !important;
   color: black;
}
```

HTML 文書作成者指定：

```
BODY{
   line-height: 1.5em;
   color: red;
}
```

採用される値：

```
BODY{
   line-height: 1.3em;
   color: red;
}
```

CSS1 および 2 の仕様書は，WWW ブラウザ標準スタイルシートに!important をつけた場合については言及していません．

ところで，指定にクラスやIDが絡んでくると，処理は複雑になります．本書では，このケースに関しての解説は省略します．

1つのシートは，いくつかの関連する指定を含んでいるのが普通です．たとえば，前景色と背景色は，互いに色が衝突しないように指定されるのが普通です．!importantはこのようなバランスを崩す可能性のある指定であり，その記述には十分な注意を払うようお願いします．

ユーザ（読者）標準スタイルシートの例

これまでのところ，Netscape NavigatorやInternet Explorerの標準的な見栄えでは，行幅に余裕がありません．行幅はスタイルシートによって自在に変更できるのですが，いまだシートを使っていないHTML文書も数多く公開されています．文書作成者がシートをつけてくれるのを待つよりも，自分で「ユーザ（読者）標準スタイルシート」を用意してしまうほうが賢いでしょう．

筆者が使っている「ユーザ（読者）標準スタイルシート」は，次のように簡単なものです．これ以上ごちゃごちゃと指定してしまうと，HTML文書作成者のシートとの兼ね合いがうまく行かなくなってしまうからです．

```
/*default.css*/
BODY{
    font-size: 11pt !important;
    line-height: 1.5;
}
```

なお，Netscape Navigator4.0には，ユーザ（読者）標準シートを設定する機能がないようです．Internet Explorer4.0には，メニューの「表示→インターネットオプション」で開かれる設定ダイアログの「全般」の中の「ユーザ補助」に設定項目があります．

HTML文書との連携

スタイルシートとHTML文書の連携に関する取り決めは，CSSの仕様ではなくHTMLの仕様の範疇に入ります．HTML4.0の仕様には，CSSだけでなく，ほかの仕組みのスタイルシートを包括した「スタイルシートとの連携方法」が記載されています（2.0，3.2の仕様のHTML文書でも，同様にLINK要素を用いてスタイルシートと連携させられると考えてもよいでしょう）．

4.3.1. 連携方式の分類

HTML4.0は，スタイルシートの連携方式を3つ想定しています．機能的な違いは「いつ有効になるのか」だけです．

▶persistent（永続）スタイルシート

常に有効になるスタイルシートです（WWWブラウザのスタイルシート機能をオフにしたときだけ，persistentシートもオフになります）．

▶preferred（おすすめ）スタイルシート

ユーザが何も指示しない場合に限り有効になるスタイルシートです．HTML文書読み込み時にはすべて有効になっていますが，特定のタイトル名のシートだけを有効にするよう切り替えることが可能です．

▶alternate（代替）スタイルシート

ユーザがとくに指示した場合に限り有効になるスタイルシートです．HTML文書読み込み時にはすべて無効になっていますが，特定のタイトル名のシートだけを有効にするよう切り替えることが可能です．

4.3.2. LINK要素で外部スタイルシートと連携

LINK要素を用いれば，独立したファイルとして保存されているスタイルシートをその文書と連携させることが可能です．ファイル名はHREF属性で指定します．シートとの連携方式は，REL（Forward Relationship）属性とTITLE属性をもちいて指定します．スタイルシートの種類はTYPE属性で（CSSであれば“text/css”と）指定します．

・シートのファイル名 ……………HREF属性
・シートとの連携方式 ……………REL属性とTITLE属性
・シートがCSSであることの明示 ….TYPE属性

▶「REL="STYLESHEET"」とする場合
TITLE属性を省略したものはpersistent（永続）スタイルシートになります．
TITLE属性を明示したものはpreferred（おすすめ）スタイルシートになります．

▶「REL="ALTERNATE STYLESHEET"」とする場合
alternate（代替）スタイルシートになります．TITLE属性を明示しなければいけません．

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>HTML4.0(W3C Recommendation 18-Dec-1997)解説</TITLE>
<LINK REL="STYLESHEET" TYPE="text/css" HREF="normal.css">
<LINK REL="STYLESHEET" TITLE="fancy" TYPE="text/css" HREF="fancy.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="compact" TYPE="text/css" HREF="small.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="compact" TYPE="text/css" HREF="mini.css">
</HEAD>
```

WWWブラウザは，読み込んだHTML文書にスタイルシートファイルが指定されている場合，そのシートも自動的に読み込みます．そして，連携方式を判断し，適切なシートだけを有効にした上で表示を調整します．

4.3.3. 有効スタイルシートの切り替え

preferred（おすすめ）およびalternate（代替）のスタイルシートには，TITLE属性によってタイトル名が指定されています．ユーザは，その中の特定のタイトル名を指示することによって，有効になるスタイルシートを切り替えることが可能です．

特定のタイトル名が指定された場合，そのタイトル名でないpreferred（おすすめ）スタイル

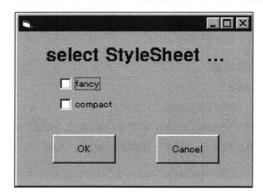
シートは無効になり，そのタイトル名のalternate（代替）スタイルシートは有効になります．なお，persistent（永続）スタイルシートは常に有効です．

```
<LINK REL="STYLESHEET" TYPE="text/css" HREF="normal.css">
<LINK REL="STYLESHEET" TITLE="fancy" TYPE="text/css" HREF="fancy.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="compact" TYPE="text/css" HREF="small.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="compact" TYPE="text/css" HREF="mini.css">
```

この例の場合，文書読み込み時にはpersistent（ここではnormal.css）とpreferred（ここではfancy.css）の2つのスタイルシートが有効になっています．ユーザの指示により「compact」が選択されれば，prefrredとalternateのうち「compact」のタイトル名を持つスタイルシート（ここではsmall.cssとmini.css）に加えてpersistentが有効になり，ほかのシートは無効になります．

なお，複数のスタイルシートに同じタイトル名を付けた場合，そのシート群は同時に有効/無効になります．

有効スタイルシートの切り替えをどのような方法で行うのかはWWWブラウザに一任されています．プルダウンメニューの一覧から選ぶ，別ウィンドウに示されるリストから選ぶなどのバリエーションが考えられます（図4-8）．

図4-8．スタイルシート選択ウィンドウ（想像図）

どのような方法をとるにせよ，CSS対応を名乗るWWWブラウザは，少なくとも指定されているスタイルシートのタイトル名一覧（ファイル名ではない）をユーザに提示できなければいけません．それにもかかわらず，Netscape Navigator4.0，Internet Explorer4.0はスタイルシート選択機能をサポートしていません．

4.3.4. 複数のスタイルシート間で宣言が衝突した場合

複数のスタイルシートが有効になっている場合には，シート間で宣言が衝突する可能性があります．衝突した場合には，単純にHEAD要素中での出現順位がもっとも後ろである宣言のみが有効になります（「4.1.4.宣言が衝突した場合の処理」を参照）．

4.3.5. 有効メディアの区別

HTML文書の再生メディアは，必ずしもコンピュータのスクリーン（ページの継ぎ目はなく，表示しきれない部分はスクロールバーによって調整する）であるとは限りません．場合によっては音声出力（スピーチ・シンセサイザー）であったり印刷（ページの物理的な区切りに制約される）であったりしても何の不思議もありません．CSS2の仕様書には，**表4-1**に示すメディアの種類が紹介されています．

表4-1．スタイルシートのメディア指定

値	説明
SCREEN	スクロール可能な（ページの区切り目の無い）コンピュータ画面
TTY	テレタイプや端末画面など，固定ピッチ文字によるコンピュータ画面
TV	家庭用TVセットのモニタ
PROJECTION	投影機
HANDHELD	小型情報機器（画面サイズや色に制限が多いもの）
PRINT	紙，あるいはページ区切りのあるコンピュータ画面
BRAILLE	点字（ページ区切りなし）
EMBOSSED	点字（ページ区切りあり）
AURAL	音声出力（スピーチ・シンセサイザー）
ALL	すべて

再生メディアが異なれば，望ましい再生スタイルも異なります．したがって，スタイルシートはメディア別に指定されているべきです．その指定は，LINK要素にMEDIA属性を指定することで実現されます（1つのスタイルシートに複数のメディアを指定する場合は，カンマ(,)で区切って並べます）．

```
<LINK REL="STYLESHEET" MEDIA="ALL" TYPE="text/css" HREF="normal.css">
<LINK REL="STYLESHEET" MEDIA="AURAL" TYPE="text/css" HREF="normal-vox.css">
<LINK REL="STYLESHEET" TITLE="fancy" MEDIA="SCREEN"
   TYPE="text/css" HREF="fancy.css">
<LINK REL="STYLESHEET" TITLE="compact" MEDIA="SCREEN, HANDHELD"
   TYPE="text/css" HREF="small.css">
<LINK REL="ALTERNATE STYLESHEET" TITLE="slow-vox" MEDIA="AURAL"
   TYPE="text/css" HREF="slow-vox.css">
```

もし，WWWブラウザが音声出力用に設定されていれば，AURALとALL以外のメディアのスタイルシートは常に無効になるでしょう．この場合はpersistent（永続）としてnormal.cssと

nomal-vox.cssが，alternate（代替）として「slow-vox（slow-vox.css）」のみが採用されるでしょう（Netscape Navigator4.0，Internet Explorer4.0はメディア選択機能をサポートしていません）．なお，MEDIA属性は省略可能ですが，省略した場合にそのシートがどのメディア用のシートとして解釈されるのかはWWWブラウザに一任されています．

4.3.6. STYLE要素で文書内にスタイルシートを記述

HTML4.0からは，HTML文書内にスタイルシートを直接記述するためのSTYLE要素が用意されています．LINK要素と同様に，TYPE属性，MEDIA属性，TITLE属性を指定できます．

```
<HEAD>
<TITLE>test</TITLE>
<STYLE MEDIA="ALL" TYPE="text/css">
    BODY{
        color: white;
        background: black;
    }
</STYLE>
</HEAD>
```

HTML4.0の仕様書には，STYLE要素におけるシートとの連携方式に関して何も明言されていません．しかし，このシートは文書の一部であることから，常に有効になる（persistentである）と思われます．なお，今のところNetscape Navigator4.0やInternet Explorer4.0はそもそもシート選択機能をサポートしていないので，どちらにせよSTYLE要素のシートを常に有効にします．

HTML3.2ではSTYLEは単なる予約語であり，具体的な取り決めは行われていませんでした．そのためTYPE属性などは定義されていません．

LINK要素とSTYLE要素の使い分け

スタイルシートは一般に独立したファイルとするのですが，STYLE要素を用いればHTML文書内に直接書き込むことも可能です．この2つはどのように使い分ければよいのでしょう．

独立したファイルとして作成したスタイルシートは，LINK要素を介して複数のHTML文書から使用できます．しかし，STYLE要素に記述したスタイルシートは，そのHTML文書からしか活用できません．したがって，**使いまわしの効くスタイルシートは独立ファイル**とし，特定のHTML文書でしか使えないような特殊な指定をしたシートはSTYLE要素に記述するのが賢い使い分けだと考えられます．

スタイルシートをコメントアウト

HTML3.2以上の仕様を解釈しないWWWブラウザは，STYLE要素の中身を単なるテキストとして表示してしまうかもしれません．W3Cは，それを避けるために，STYLE要素の中身をHTML文書としてコメントアウトするよう提案しています．

```
<STYLE TYPE="text/css">
<!--
    BODY{
        font: 400 medium serif;/* 太さ400，サイズ普通，ひげ付き文字 */
        text-align: left;
    }
-->
</STYLE>
```

なお，あくまでも「HTML文書としてコメントアウト」ですから，<!--～-->です．/*～*/だと，「CSSとしてコメントアウト」になり，宣言がすべて無効になります．

4.3.7. 開始タグ中にスタイル宣言を埋め込む

HTML4.0からは，BODY要素に含まれる（SCRIPTとPARAMを除く）すべての要素にSTYLE属性があり，属性値としてスタイル宣言を記述できます．この方法は，例外的・局所的なスタイル宣言を追加するために利用できます．

```
<H3 STYLE="color : red; font-size: 1.5em">3.1.1.HTMLに関して</H3>
<H4 STYLE="text-align: center">/HTMLとは/</H4>
<P>HTML (hypertext markup langauge)とは，…</P>
```

文法的には，LINK要素やSTYLE要素を使用せずにすべてのスタイルをSTYLE属性で指定することも可能です．しかし，筆者はスタイル指定がHTMLマークアップ内部に入り込むのは望ましくないと考えています．強制ではないのですが，STYLE属性による局所指定は最小限にとどめるよう希望します．

CSS properties for visual media in

5章では，CSSが規定する個々のプロパティ（property）に関する解説を行います．5章を暗記する必要はありません．5章は「辞書」としてお使いください．なお，プロパティの総合的な使い方は6章に譲ります．

5-1 リファレンスの読み方

CSS2では，以下の特徴に着目して再生メディアを分類します．

着眼点	区分
大区分	画像(visual)，音声(aural)，接触(tactile：点字など)
ページ区切り	ある(continuous)，ない(paged)
画素	グリッド(grid：文字端末など)，ビットマップ(bitmap)
対話性	ある(interactive)，ない(static：印刷物など)

多くのプロパティは複数のメディアで共通ですが，中には「音声再生専用のプロパティ」なども存在します．なお，CSS1が用意したのは，「visualでcontinuousのみ，あるいはpagedと共通」のプロパティです．CSS2で追加されたのは「aural」「visualでpaged専用」のプロパティです．そして，読者の興味の大部分はCSS1相当の区分に集中しているものと思われますし，実装が進んでいるのものこの区分です．そのような都合から，本章で詳細に取り扱うのは「visual（主にCSS1相当)」とし，その他のCSS2追加プロパティについては7章「new feature in CSS2」にまわします．

なお，本書のプロパティ定義欄の記述は，W3Cの仕様をもとに，読みやすくするためにカッコなどを追加したものです．表記が変更されていても内容は変更されていませんので，あらかじめご了承ください．

5.1.1. 値定義の読み方

この章では，次の形式で各プロパティを紹介していきます．

margin-left　左マージン

値	<一般サイズ> | <パーセント> | auto /*auto …自動処理 */
初期値	0
適用対象	全要素
継承される？	no
パーセントの意味	親要素のwidthに対する割合
マイナス値	可

値の定義では，次の特殊な記号を使用します．

◆原則

・指示した順に，一度だけ記述

・キーワード…基本的にそのまま記述．カンマ（,）やスラッシュ（/）などもキーワードになりうる．

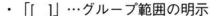
・「/*　*/」…筆者が付けたコメント

◆選択回数の明示

・「｜」…グループから一つだけ記述

font-style（斜体）

値	normal ｜ italic ｜ oblique
正しい解釈：	font-style: normal;
	font-style: italic;
間違った解釈：	font-style: normal italic;　**/*二つは書けない*/**
	font-style: "normal";　**/*キーワードには""を付けない*/**

・「[　]」…グループ範囲の明示

・「‖」…グループから，順不同で複数記述可能，ただし同じキーワードは一度だけ

text-decoration　下線など

値	none ｜ [underline ‖ overline ‖ line-through ‖ blink]
正しい解釈：	text-decoration: none;
	text-decoration: blink underline;
間違った解釈：	text-decoration: none underline;　**/*none は｜｜のグループではない*/**
	text-decoration: underline overline underline;　**/*2度は書かない*/**

◆値の説明の明示

・「<　>」…書くべき値の説明（本文中で説明）．パーセント，url，色など．

color　前景色（文字色）

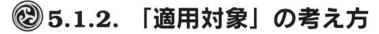
値	<色>
正しい解釈：	color: #fff;
	color: rgb(0, 0, 255);
	color: red;
間違った解釈：	color: <red>;　**/*<>はいらない*/**

◆記述回数の明示

- 「*」…0回以上繰り返し
- 「?」…0か1

font　フォント関連の一括指定

値	[<font-style> || <font-variant> || <font-weight>]? <font-size> [/<line-height>]? <font-family> /*line-heightの直前のスラッシュは，そのままスラッシュとして記述*/ /*font-sizeとfont-familyだけは必須*/
正しい解釈：	font: 700 15pt serif; font: italic 10pt/1.5 sans-serif;
間違った解釈：	font: normal serif; **/*font-sizeは省略できない*/** font: 10pt 1.3 serif; **/*line-heightの直前には「/」が必須*/**

5.1.2.「適用対象」の考え方

どの要素にも，すべてのプロパティを指定可能ですが，全部が適用されるとは限りません．たとえば，text-align（水平位置そろえ）はインライン系要素（STRONG要素など）には適用されません（なぜなら，適用しようがないからです）．このように，CSSでは，対象要素がブロック系要素であるかインライン系要素であるかによって，プロパティを適用するかどうか判断します．

なお，その所属の初期設定はHTMLの仕様で定められているとおりですが，スタイルシートのdisplayプロパティ（5.8節）によって，表示形式だけは変更可能です．したがって，厳密には，各プロパティが適用されるか否かはdisplayプロパティの値で判断されます．

CSSでは，特殊なインライン系要素として「置換要素」を想定しています．置換要素とは，IMG要素など，ソース中の文字列が画像などに置きかえられた後に表示されるものです．プロパティによっては，「置換要素」の場合には意味合いが変化します（たとえばwidthプロパティ：5.7節）．

5.1.3. CSS1とCSS2の違いについて

CSS2は，CSS1の上位互換の仕様として作成されました．すなわち，CSS1のプロパティはすべてCSS2でも採用されています．したがって，とくに何も明示しない限り，この章で紹介するプロパティは共用です．しかし，実際には一部のプロパティについて改変が行われました．その場合は，必ずその旨を本文中に明示しています．

また，CSS2では，すべてのプロパティの値に「inherit」を導入しました（4.2を参照）．これはCSS1には存在しないキーワードであるため，本書ではあえてinheritを値リストに表示しませんでした．これについてはご注意ください．

5-2 「一般サイズ」単位の解説

いくつかのプロパティに共通して使われるサイズの単位に関して，あらかじめここで説明しておきます．

5.2.1. 絶対単位

絶対単位とは，サイズを絶対尺度で指定する単位です（**表5-1**）.

表5-1．CSSにおける絶対単位

単位	意味	備考
mm	ミリメートル	
cm	センチメートル	1cm = 10mm
in	インチ	1in = 25.4cm
pt	ポイント	72pt = 1in
pc	パイカ	組版用語，1pc = 12pt

（Lie and Bos(1)より）

5.2.2. 相対単位

「現在の文字サイズ」を意味するem（エム），ex（エックス）と，「画面の解像度に依存する」px（ピクセル）は，状況によってサイズの変動する単位，すなわち相対単位です．

『em』はもともとは組版用語で，アルファベットMのサイズ，転じて「文字を描画するのに必要な**高さ**」を意味しています．CSSでは**『その要素における』フォントサイズそのもの**として扱われます．たとえば，「line-height: 1.5em」は，フォントサイズが14ptならば21ptに，10ptならば15ptに設定されます．

font-sizeプロパティのみ，例外的にemユニットを「親要素のフォントサイズ」として取り扱います．

横方向の指定にもemユニットは活用されますが，その際でもemユニットは「フォントの高さ」に相当します．文字は必ずしも正方形ではないのでご注意ください．日本でのemの組版における意味合いは「全角文字の幅」となっています（新フォント関連用語集：財団法人　日本規格協会）．この訳だと「縦幅」なのか「横幅」なのか明確ではないため，本書は「高さ」と表現しました．もっとも，CSSではあくまでも『その要素における』フォントサイズそのものとして扱われます．

『ex』はその要素のフォントサイズにおける「x-height」の大きさです（**図5-1**）．フォントファミリーが異なれば，同じフォントサイズでもex値は変ります．「x-height」の値は，フォントデータそのものに含まれています（「x-height」という概念はラテンアルファベット圏のものであり，日本語文字では利用されません）．

図5-1．x-height

pxはピクセル，すなわち画面に表示できる最小の画素のサイズです．同じサイズのディスプレイでも，解像度設定（640×480とか，800×600とか）によってピクセルのサイズは変化します．

5.2.3. パーセント

単位「%」は，サイズをパーセントで指定するものです．ただし，「何に対するパーセントなのか」はプロパティによって異なります．たとえば，widthプロパティでは表示領域の横幅に対するパーセントを，font-sizeプロパティではその親要素の文字サイズに対するパーセントとなります．詳しくは各プロパティの単元で説明します．

5.2.4. 比率指定の継承に関する注意

emユニットと%による指定が継承される場合は，**相当する絶対値に変換された後に**継承されます．次の例の場合，P.STARTのfont-sizeは12pt × 1.2=14.4ptであり，その子要素は14.4ptを継承します．

ところが，相対値として継承されるとすると，次の例ではP，CITE，STRONGの順に文字が大きくなってしまいます．

```
<STYLE TYPE="text/css">
BODY{font-size: 12pt}
P.START {font-size: 1.2em}
</STYLE>
<BODY>
<P CLASS="START">最近読んだ書籍<CITE>「ミュージックマガジン
<STRONG>10月号</STRONG>」</CITE>では，興味深い特集が組まれていた．</P>
</BODY>
```

（この文脈では，CITEはPから指定を継承し，STRONGはCITEから継承します．）

継承が「計算結果」の場合の継承結果：

```
CITE{font-size: 14.4pt}
STRONG{font-size: 14.4pt}
```

継承が「相対性」の場合の継承結果：

```
CITE{font-size: 17.28pt  /* 14.4*1.2=17.28pt */}
STRONG{font-size: 20.736pt /*  17.28*1.2=20.736pt */}
```

5-3 フォント系

▶ 単体指定

font-family ················フォントファミリー
font-size ··················サイズ
font-weight ················太さ
font-style··················斜体
font-variant················スモールキャピタル
line-height ················行幅

▶ 一括指定

font

▶ CSS2 追加分

font-stretch················横方向の引き延ばし
font-size-adjust ············サイズ修正

```
BODY{
    font: 12pt/1.5 "Times Roman", "Times", serif;
}
STRONG{
    font-style: italic;
    font-size: 1.2em;
}
H1{
    line-height: 1;
}
```

5.3.1. 単体指定

※ font-family フォントファミリー

値	[[< ファミリー名 >	< 系統名 >],]* [< ファミリー名 >	< 系統名 >]
初期値	実装依存
適用対象	全要素
継承される？	yes

（実装依存とは，WWWブラウザが適切な値を選択する，という意味です）

専門用語の厳密な解説はしませんが，「フォントファミリー名」とは"Times New Roman"や"Century"のような名前です（ファミリー名は定義済みのキーワードではありませんので，クォーテーション（""あるいは''）で囲みます）．

このプロパティには複数の値を指定できます．複数の指定がある場合，WWWブラウザは前者から順に所有するファミリーを探し，最初に見つけ出したものを採用します．

```
font-family: "Times Roman", "Times", serif;
/* まず"Times Roman"を探し，無ければ"Times"を，それも無い場合は serif を */
```

また，“ファミリー名”ではなく，**ファミリーの一般的な性質を示唆する“系統名”**（generic-family）でもファミリーを指定できます（**表 5-2**）．系統名で指定した場合，WWWブラウザが存在するファミリーの中から適切なものを選択します．一般に，文書の読み手がどんなファミリーをもっているのか推測することは不可能なため，系統名を利用する方が望ましいでしょう（**系統名は**定義済みのキーワードであるため，**クォーテーションでは囲みません**．囲ってしまった場合は，"serif"などという特定の名前のファミリーを指定したことになります）．

表 5-2. 系統名とその説明

系統名	意味	具体的なファミリーの例
serif	ひげ付き	"Times Roman"　明朝系
sans-serif	ひげ無し	"Arial"　ゴシック系
monospace	等幅	"Courie"
cursive	手書き風	*"Zapf Chancery"*　楷書
fantasy	装飾付き	"Goudy"　POP文字

※ font-size サイズ

値	xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | < 一般サイズ > | < パーセント >
初期値	medium
適用対象	全要素
継承される？	yes
パーセントの意味	親要素の font-size に対する割合
マイナス値	不可

「xx-small ～ xx-large」は文脈に依存しない固定サイズ指定で，この順に大きくなります．ただし具体的なサイズは WWW ブラウザの設定に一任されます（CSS2 では，それぞれの関係は，小さいほうから大きいほうへ「1.2 倍程度」だとされています）．

「larger」「smaller」は相対指定で，親要素のサイズよりも一段階大きく（小さく）なります．この「一段」がどの程度かは WWW ブラウザに一任されますが，基本的には「medium」が「large」に（「small」に）なるのと同程度の変化だと思われます．なお，継承されるときは相当する絶対値として継承されます．

「大きさ」に上限はありません．xx-large よりも大きなサイズになることも可能です．
「小ささ」は 0 が下限です．

相対指定のススメ

相対指定はインライン系要素にとってはとくに重要です．たとえば，STRONG 要素に対し

```
STRONG{font-size:1.2em}
```

と指定しておけば，フォントサイズが 15pt の H1 要素中に現われた場合でも，10pt の P 要素中に現われた場合でも，常に「親要素のフォントサイズの 1.2 倍」という関係を自動的にキープできます．

同様に，

```
STRONG{font-weight: bolder}
```

と指定しておけば，太さ 700 の H1 要素中に現われた場合でも，400 の P 要素中に現われた場合でも，常に「親要素よりも太い」という関係がキープできます．

※ font-weight 太さ

値	normal	bold	100	200	300	400	500	600	700	800	900	bolder	lighter /* 100が最も細く，900が最も太い．*/ /*nomal = 400，bold = 700*/ /*250などは指定できません．*/
初期値	normal
適用対象	全要素
継承される？	yes

「100～900」は太さの絶対サイズで，これ以外の値（たとえば250）は指定できません．ただし，実際の表示として9段階の太さが区別できるかどうかはWWWブラウザおよびフォントに一任されます．

「bolder」「lighter」は相対指定で，親要素の指定よりも一段階太く（細く）なります．この「一段」はWWWブラウザに任されますが，「400」が「500」に（「300」に）なるのと同程度の変動が期待できます（上限は900，下限は100です）．なお，継承されるときは相当する絶対値として継承されます．

※ font-style 斜体

値	normal	italic	oblique
初期値	normal
適用対象	全要素
継承される？	yes

italicもobliqueも日本語では同じ「斜体」と受け取られがちですが，ラテンアルファベットのファミリーによっては，この2者が区別されます（**表5-3**）．

表5-3．italicとobliqueの意味

値	表示例	説明
italic	*Aa*	専用斜体．ノーマルとは文字形状まで異なる．
oblique	*Aa*	単純斜体．ノーマルが単に傾いた文字形状．

※ font-variant スモールキャピタル

値	normal	small-caps
初期値	normal
適用対象	全要素
継承される？	yes

small-capsを指定すると，“小文字”で書かれたテキストがスモールキャピタル文字に調整されます（**図5-2**）．なお，この概念は日本語文字には存在しません．

図5-2. スモールキャピタルの例

```
<P><SPAN CLASS="FAMILYNAME">Zappa</SPAN>, Frank
```

ZAPPA, Frank

※ line-height　行幅

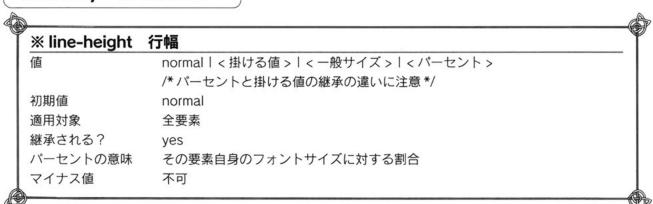
値	normal｜<掛ける値>｜<一般サイズ>｜<パーセント> /* パーセントと掛ける値の継承の違いに注意 */
初期値	normal
適用対象	全要素
継承される？	yes
パーセントの意味	その要素自身のフォントサイズに対する割合
マイナス値	不可

行幅（行送り）は，**上下に均等**に割り振られます．文字そのものの幅も含んだ値を指定してください（**図5-3**）．

図5-3. line-height

{line-height: 1.4em}　の場合

ABCkjp

0.2em
1.0em
0.2em

<掛ける値>は，「その要素のフォントサイズの何倍を行幅にするか」を無単位の実数で記述します．

```
line-height: 1.4;
```

<掛ける値>の効果は，emユニットとまったく同じです．ただし，継承のされかたが異なります．5.2節で説明したとおり，emおよびパーセントは相当する絶対値に変換された後に継承されます．しかし，line-heightの<掛ける値>は常に**相対値として**継承されます．この2つは，望む効果に応じて使い分けなければなりません（**図5-4**）．

・em，%を用いた場合…子要素の（強調などで）文字サイズが変っても，行幅は同じ
・<掛ける値>を用いた場合…子要素文字サイズが変れば，行幅もかわる

図 5-4. 行幅の継承の違い：em, 掛ける値

全て同じサイズ	文字サイズにかかわらず一定	文字サイズに相対的に変化
Test Test Test Test	Test Test Test Test	Test Test Test Test
Test Test Test Test	Test Test Test Test	Test Test Test Test
Test Test Test Test	Test Test Test Test	Test Test Test Test
Test Test Test Test	Test Test Test Test	Test Test Test Test
Test Test Test Test	Test Test Test	Test Test Test
Test Test Test Test	Test Test Test Test	Test Test Test Test
Test Test Test Test	Test Test Test Test	Test Test Test Test
		Test

5.3.2. 一括指定

一括指定プロパティ（ショートハンド・プロパティ）は，関連するプロパティを一括して指定するためのプロパティです．

※ font

値	[<font-style> || <font-variant> || <font-weight>]? <font-size> [/<line-height>]? <font-family> /*line-height の直前のスラッシュは，そのままスラッシュとして記述 */ /*font-size と font-family だけは必須 */
初期値	個々のプロパティの初期値
適用対象	全要素
継承される？	yes
パーセントの意味	個々のプロパティのとおり

一括指定プロパティの場合，値指定を省略した個々の部分は，**「初期値を明示した」と解釈されます**のでご注意ください．たとえば，

```
{
    line-height: 1.5;
    font: 12pt sans-serif;
}
```

は，

```
{
    line-height: 1.5;
    font: nomal nomal nomal 12pt/nomal sans-serif;
}
```

として解釈されます．そのため，font プロパティ以前に指定していた line-height の指定は無効になります．

5.3.3. CSS2 追加分

※ font-stretch 横方向の引き延ばし

値	normal	wider	narrower | ultra-condensed	extra-condensed	condensed | semi-condensed	semi-expanded	expanded | extra-expanded	ultra-expanded	inherit
初期値	normal
適用対象	全要素
継承される？	yes
マイナス値	不可

font-size（フォントの高さ）に対する**横の割合**の指定をするものです（**図5-5**）．通常がnomal，ultra-condensedがもっとも狭く，ultra-expandedがもっとも広くなります．なお，具体的な比率は規定されていません．

widerとnarrowerは，親要素に対する相対指定です．

図5-5. font-stretchの適用例

condensed　normal　expanded

※ font-size-adjust サイズ修正

値	<number>	none	inherit /*none ＝修正しない*/
初期値	none
適用対象	全要素
継承される？	yes
マイナス値	不可

同じfont-sizeであっても，font-familyが異なればサイズが違うように感じることがあります．その原因は，x-heightの違いにあります．そこで，font-sizeをx-heightの違いに合わせて修正するために，font-size-adjustプロパティを用います．

font-size-adjustは，font-familyを複数指定した場合にのみ利用します．font-size-adjustの値には，font-familyで指定したファミリーのz値（すなわち，font-size ÷ x-height）を設定します．WWWブラウザは，実際に選択されたファミリーのz値を用いて，以下のようにfont-sizeを修正します（**図5-6**）．

修正後のfont-size＝font-size∗font-size-adjust/採用されたファミリーのz値

図5-6. フォントサイズ修正結果例

Verdana: **1**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Comic Sans MS: **1.07**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Trebuchet MS: **1.09**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Georgia: **1.16**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Myriad Web: **1.2**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Minion Web: **1.23**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Times New Roman: **1.26**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Gill Sans: **1.26**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Bernhard Modern: **1.45**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

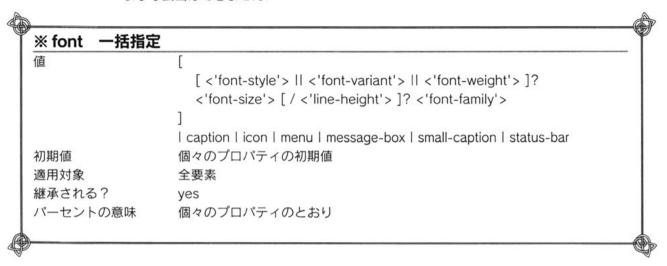
Caflisch Script Web: **1.57**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

Flemish Script: **2.07**
xylophone synergy diaphragm partially hydrogenated
vegetable shortening or lengthening or resting

（CSS2仕様書15.2.4より転載．図中の数字は，フォントサイズを何倍したかを表している．それぞれでフォントサイズは異なっているが，見た目の印象は同じサイズになっている．）

このプロパティはフォントデータにZ値が含まれていること，WWWブラウザがその値を判別できることを前提にしています．今のところ，一般向けのWWWブラウザではこのような芸当はできません．

※ font　一括指定

値	[　[<'font-style'> || <'font-variant'> || <'font-weight'>]? 　<'font-size'> [/ <'line-height'>]? <'font-family'>] | caption	icon	menu	message-box	small-caption	status-bar
初期値	個々のプロパティの初期値
適用対象	全要素
継承される？	yes
パーセントの意味	個々のプロパティのとおり

CSS2ではfontプロパティの指定方法として，「システムフォント設定を指定する」方法が追加されました（なお，CSS2での追加プロパティが指定内容に組み込まれていないことに注意してください）．

カッコ[ ]の位置に注目すると分かりますが，指定は「CSS1と同じようにする」と「システムフォント名を記述する」の**どちらか一方**しか選べません．システムフォントを指定した場合は，フォントサイズなどを細かく指定できないことになります．もし調整を行いたいならば，**fontプロパティより後で**font-sizeプロパティなどを活用してください（一括指定は，それまで行われた指定をすべて上書きしてしまいます．したがって，「後に」宣言しなければなりません）．

```
font: caption;
font-size: 1em; /*上書き指定*/
```

5-4 色と背景効果系

▶ 単体指定

color ……………………前景色（文字色）

background-color …………背景色

background-image …………背景画像

background-repeat…………背景画像の繰り返し制御

background-position………背景表示位置

background-attachment……背景スクロール

▶ 一括指定

background

▶ CSS2 追加分

システムカラーという概念

```
BODY{
    color: #fff;  /*white*/
    background: #000 url(back1.gif);  /*black, image*/
}
STRONG{
    color: red;  /* #f00, rgb(255, 0, 0)*/
}
H1{
    background: #faa url(heading.gif);
    /*要素ごとに背景を変えられる*/
}
```

5.4.1. 単体指定

※ color　前景色（文字色）

値	<色> /*表5-4*/
初期値	実装依存
適用対象	全要素
継承される？	yes

表5-4. 色の指定方法

記述	意味
#RGB	16進1桁ずつ… #820は#882200に相当
#RRGGBB	16進2桁ずつ
rgb(0-255, 0-255, 0-255)	0-255で
rgb(100%, 100%, 100%)	0-100%で
white など	定義済みの名前（**表5-5**）

色指定に関して

数値と色の詳細な関係は，ソフトウェアなどで確認するべきでしょう．たとえば，Windows95のシステムカラー・パレットでも確認できます．なお，#RGB形式による指定結果一覧を本書のWWWサポートページにて公開します．

http://www.gihyo.co.jp/css/

ただし，すべてのWWW利用者がフルカラー表示のモニタを利用しているとは限りません．画像で利用している色も含んで256色以上を指定する場合は，とくに**カラーバランスにご注意**ください．

表 5-5. 定義済みカラーテーブル

定義済みの名前	値	近似値	備考
red	#ff0000	#f00	R が最大，G と B は 0，したがって赤.
lime	#00ff00	#0f0	ライム…明るい緑
bule	#0000ff	#00f	青
black	#000000	#000	黒
gray	#808080	#888	灰色＝（黒＋白）÷ 2
silver	#c0c0c0	#ccc	薄い灰色
white	#ffffff	#fff	白
maroon	#800000	#800	栗色
green	#008000	#080	通常の緑
navy	#000080	#008	紺
purple	#800080	#808	紫＝（赤＋青）÷ 2
olive	#808000	#880	オリーブ
teal	#008080	#088	青緑
yellow	#ffff00	#ff0	黄
fuchsia	#ff00ff	#f0f	濃い紫
aqua	#00ffff	#0ff	水色

CSS1 仕様書は，この「色の名前」は「Windows VGA palette で定義されている」としていますが，CSS2 では「HTML4.0 仕様書で定義されている」と変更されました.

※ background-color　背景色	
値	< 色 > | transparent
初期値	transparent
適用対象	全要素
継承される？	no

4 章で説明したとおり，通常は「transparent（透明)」で，親要素の背景がそのまま使われます.

※ background-image　背景画像	
値	<url> | none
初期値	none
適用対象	全要素
継承される？	no

URL は「url(」と「)」で囲って示します．絶対 URL でも相対 URL でも構いません．相対 URL は，HTML 文書からではなく “スタイルシートファイルからの” 相対 URL として記述します.

一般形式：

```
url(ここにURLを書く)
```

例：

```
background-image: url(http://www.foo.com/bar/img/dog.gif);
background-image: url(cat.gif);
```

スタイルシートを使おう！

スタイルシートを用いない場合，文字色/背景色/背景画像はBODY要素の属性で指定します（HTML3.2，4.0-Transitional）．しかし，この方法では本文全体への一括指定しかできません．

ところが，スタイルシートを用いれば，任意の要素に対して文字色・背景色・背景画像を指定できます．すなわち，H1要素だけの背景画像，H2要素だけの背景画像などを別々に指定できるのです．このことは大変単純なことがらですが，「見栄え指定の進歩」という意味では驚異的だと思いませんか？

HTMLだけで無理にスタイルを指定しようとしている人は，いつまでたってもこの利点を享受できません．このことを友人みんなにふれてまわってください．

「スタイルシートを使おう！　だって，こんなにすごいんだぜ」

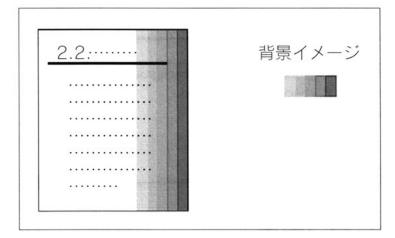
※ background-repeat 背景画像の繰り返し制御

値	repeat I repeat-x I repeat-y I no-repeat /* 水平垂直 | 水平のみ | 垂直のみ | 繰り返さない */
初期値	repeat
適用対象	全要素
継承される？	no

※ background-position　　背景表示位置

値	[[<パーセント> I <一般サイズ>] [<パーセント> I <一般サイズ>]?] I [[top I center I bottom] II [left Icenter I right]] /* 垂直位置　水平位置 */ /* 水平垂直を同じにする場合は，一方だけ指定すればよい */
初期値	0% 0%　/*top left*/
適用対象	ブロック系要素と置換要素
継承される？	no
パーセントの意味	要素自体の表示サイズ（width や height）に対する割合

背景画像の表示開始位置を設定します．もしrepeat-yかつbackground-positionがrightであれば，その要素の右端に表示され，垂直方向に繰り返されます（**図5-5**）．

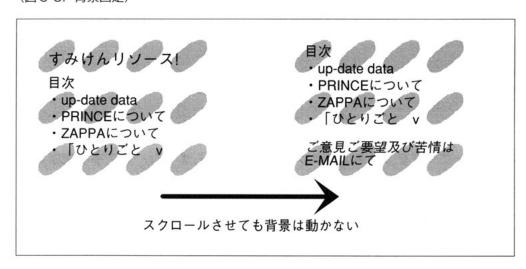
図 5-7. background: repeat-y right

※ background-attachment　背景スクロール

値	scroll | fixed
初期値	scroll
適用対象	全要素
継承される？	no

指定がfixedの場合は，前景文字などをスクロールさせても，背景画像はスクロールしません（**図5-8**）.

（図 5-8. 背景固定）

5.4.2. 一括指定

※ background

値	<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>
初期値	個々のプロパティの初期値
適用対象	全要素
継承される？	no
パーセントの意味	個々のプロパティのとおり

5.4.3. システムカラーという概念

CSS2では，<色>の値として「システムカラー」を導入されました．

近代的なGUIを備えたシステムでは，メニューやボタンやウィンドウ枠の色などを，個々のアプリケーションが管理するのではなく，システムが一括管理するようにしています．この「OSが管理する色」を「システムカラー」といいます．ユーザは「システムカラー」の設定を変更するだけで，すべてのアプリケーションの表示色を変更できるわけです．システムカラーの概念は，WindowsやMac，はてはJava VM（仮想マシン）にも存在します（たとえば，Windows95では「コントロール」の「画面」の「デザイン」の各項目を変更してみてください）．

スタイルシートからシステムカラーを指定できれば，「読者が普段慣れ親しんでいる色」を指定できることになります．その一方で，シート記述時には表示色が想像できないわけですから，扱いは難しいかもしれません．

```
BODY{
    color: WindowText;
    background: Window;
}
```

CSS2は，**表5-6**のキーワードを「システムカラー」用に用意しました．大文字小文字は区別されませんが，原文と同じように記述するほうが分かりやすいでしょう．

表5-6. CSS2におけるシステムカラー

名前	対応する部分
ActiveBorder	アクティブなウィンドウ枠の色
ActiveCaption	アクティブなウィンドウ（のタイトルバー）の色
AppWorkspace	MS-WORDのような「子ウィンドウ」を持つウィンドウ（MDI）の背景色
Background	デスクトップの背景色
ButtonFace	立体的なボタンなどの地の色
ButtonHighlight	選択（反転，ハイライト表示）された立体的なボタンなどの色
ButtonShadow	立体的なボタンなどの影の色
ButtonText	ボタンの文字色
CaptionText	ラベルやタイトルバーなどの文字色
GrayText	「使用不可（グレイアウト）」になっている文字色
Highlight	選択リストなどの選択されている部分の色
HighlightText	選択リストなどの選択されている部分の文字色
InactiveBorder	アクティブでないウィンドウの枠の色
InactiveCaption	アクティブでないウィンドウ（のタイトルバー）の色
InactiveCaptionText	アクティブでないウィンドウ（のタイトルバー）の文字色
InfoBackground	ツールチップ（バルーンヘルプ）の背景色
InfoText	ツールチップ（バルーンヘルプ）の文字色
Menu	メニューの背景色
MenuText	メニューの文字色
Scrollbar	スクロールバーのバーの色
ThreeDDarkShadow	立体表示要素の暗い影の色
ThreeDShadow	立体表示要素の影の色
ThreeDLightShadow	立体表示要素の明るい影の色
ThreeDFace	立体表示要素の地の色
ThreeDHighlight	選択された立体表示要素の色
Window	ウィンドウの背景色
WindowFrame	ダイアログボックスなど（筆者注：borderとframeの違いが不明）
WindowText	ウィンドウの文字の色

5-5 テキスト属性系

▶ 単体指定

word-spacing ･････････････単語間隔
letter-spacing ･････････････文字間隔
text-decoration ･････････････下線など
vertical-align ･････････････垂直位置（上付き下付き）
text-align ･････････････水平位置そろえ（右そろえ，左そろえ）
text-transform ･････････････大文字小文字
text-indent ･････････････行頭インデント

▶ 一括指定

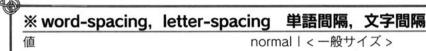
なし

▶ CSS2 追加分

text-shadow ･････････････影文字

```
BODY{ text-align: left;}
P{ text-indent: 1em;}
STRONG{
    word-spacing: 1.5em;
    text-decoration: underline overline;
}
```

5.5.1. 単体指定

※ word-spacing，letter-spacing　単語間隔，文字間隔

値	normal｜<一般サイズ> /*nomal = 0*/
初期値	normal
適用対象	全要素
継承される？	yes
マイナス値	可

word-spacingは「単語間隔」，letter-spacsingは「文字間隔」です．単語間隔はラテンアルファベット圏の概念で，単語を区切る空白などの幅になります．

指定値は「標準に対する**追加の幅**」であり，マイナス値を指定すれば文字は重なり合います．

これらのプロパティは，一般には文字の読みやすさの調整に使いますが，語句を目立たせるために極端な指定をしてもかまいません（STRONG要素など）（**図5-9**）．

図5-9．単語間隔と文字間隔

```
Word-space:
            This is NORMAL ex.
            This is    WIDE    ex.
            This isNARROWex.

letter-space:
            This is NORMAL ex.
            This is P L U S ex.
            This is MINUS ex.
```

※ text-decoration　下線など

値	none｜[underline ‖ overline ‖ line-through ‖ blink] /* 下線 ‖ 上線 ‖ 取り消し線 ‖ 点滅 */
初期値	none
適用対象	全要素
継承される？	no，しかし「透明」処理される

「無し」と指定するか，各種の指定を組み合わせた指定（「text-decoration: underline blink」など）が可能です．指定は継承されませんが，「透明」処理されるために一見継承されたように見えます．「継承されない」の意味は，「下線に更に下線が付くことはない」という意味だと考えても良いでしょう．

※ vertical-align　垂直位置（上付き下付き）

値	baseline | sub | super | text-top | middle | text-bottom | <パーセント> | top | bottom
初期値	baseline
適用対象	インライン系要素
継承される？	no
パーセントの意味	要素自身の'line-height'に対する割合で，プラスは上，マイナスは下
マイナス値	可

親要素に対する垂直位置関係を指定します．しかし，インライン系要素にしか適用されないため，一般には「直前の語句（単語）」に対する垂直位置関係になるでしょう．単純に言えば「上付き下付き」ですが，かなり複雑な指定が可能です（**図5-10**）．

いわゆる乗数 x^2 のような効果を得たい場合は，super 指定に加えてフォントサイズを小さくする必要があります．

図5-7. 垂直位置そろえ

※ text-align　水平位置そろえ（右そろえ，左そろえ）

値	left I right I center I justify /* 左端｜右端｜中央寄せ｜両端そろえ（図 5-11）*/
初期値	実装依存
適用対象	ブロック系要素
継承される？	yes

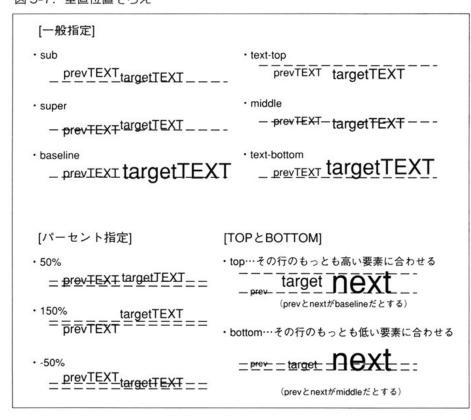
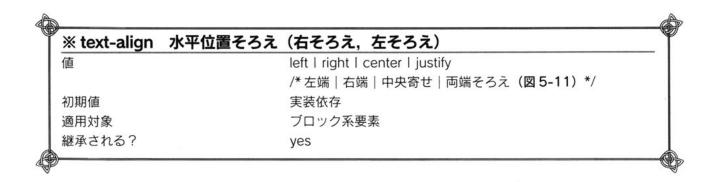
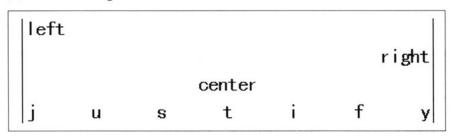



図 5-11．text-align

水平位置そろえは，ブロック系要素にしか指定できません．

CSS 仕様書は見栄え効果を指定しますが，その効果を得るための方法は指定しません．WWW ブラウザは，ピクセル単位で文字位置を整形しても，単に余分な空白文字を挿入することで表示をおおまかに調整するだけでも，その他の方法を採用しても自由です．

CSS2 では，TABLE 用の text-align として，属性値に任意の文字列を指示できるようになりました．具体的には，「text-align: "."」として**小数点そろえ**を実現するために活用されます．

※ text-transform　大文字小文字

値	capitalize I uppercase I lowercase I none /* 単語の頭だけ大文字｜大文字｜小文字｜変化無し */ /*Capitalize I UPPERCASE I lowercase I none */
初期値	none
適用対象	全要素
継承される？	yes

ソース中の大文字小文字記述を，表示時に変更するための指定です．

CSS1 仕様書には，日本語文字のように大文字小文字の区別の無い文字の場合，どの指定も none として解釈されると記されています．

※ text-indent　行頭インデント

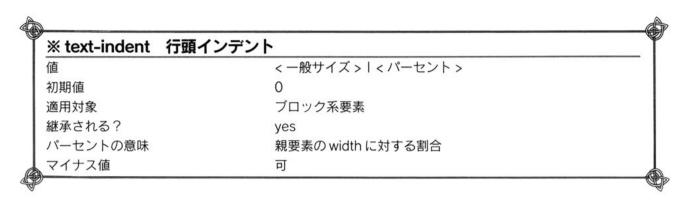
値	<一般サイズ> | <パーセント>
初期値	0
適用対象	ブロック系要素
継承される？	yes
パーセントの意味	親要素のwidthに対する割合
マイナス値	可

text-indentは，ブロックの始めの一行にのみ適用されるインデントです．マイナスのインデントも可能です（**図5-12**）．

図5-12．インデント無し，プラス，マイナス

5.5.2. CSS2追加分

※ text-shadow　影文字

値	none | [<color> || <length> <length> <length>? ,]* [<color> || <length> <length> <length>?] /* 色　水平オフセット値　垂直オフセット値　ぼかし範囲 */
初期値	none
適用対象	全要素
適用対象	ブロック系要素
継承される？	no （処理的にはtransparentと受け取るべきか？）
マイナス値	可（横は右，縦は下が「正」）

この「影文字」とは，同じ文字をすこしずらして表示する効果のことです（**図5-13**）．任意の数の「ずらした文字」を表示できます．「水平・垂直のオフセット値」とは，元の文字に対する影になる文字の移動量です．オプション指定として，影文字に「ぼかし効果」を与える指標「ぼかし範囲」を指定できます．「ぼかし範囲」を「3pt」とした場合，元々の文字の回りに3ポイント程度の「ぼかし」が表示されます．

図5-13．影文字の例

文字に影を付ける
少しずらして文字を描写すれば，影がついたような感じになります。

CSS2の仕様書には，オフセット値を0にしてぼかし範囲を大きく取ることで面白い効果を得る例が紹介されています（**図5-13**）．しかし，一方で「このプロパティは，:first-letter疑似要素など特殊な部分でのみ使用してほしい」と明記されています．

図5-14．凝った影文字

ECLIPSE

```
{
    background: white;
    color: white;
    text-shadow: black 0px 0px 5px;
} (CSS2 仕様書 16.3.2 より転載)
```

仕様と現実の違い

このtext-shadowプロパティの「ぼかし」効果は，もし実現すれば強力ですが，これまでのWWWブラウザの機能拡張の方向を見る限り，しばらくの間サポートされないのでは無いかと思われます（私見です）．

ほかのプロパティに関しても同様ですが，W3Cの仕様書に書かれていることが**常にサポートされるとは限りません**．事実として，既にCSS1仕様公開から1年以上が経ちますが，いまだにCSS1を「完全に」サポートしたWWWブラウザは存在しません．Microsoft社もNetscape Communications社も「当社はCSSをサポートする」と宣言しているにもかかわらず，これが現実です．仕様と現実は別のものであることには，十分注意してください．

5-6 ボックス系

▶ **単体指定**

margin-top, margin-right, margin-bottom, margin-left

padding-top, padding-right, padding-bottom, padding-left

border-top-width, border-right-width, border-bottom-width, border-left-width

▶ **一括指定**

margin ････････････････････マージン

padding ･･･････････････････パディング

border-width ･･････････････ボーダー幅

border-color ･･････････････ボーダーの色

border-style ･･････････････ボーダーの形

border, border-top, border-right, border-bottom, border-left

▶ **CSS2 追加分**

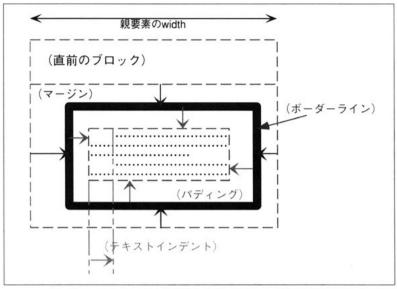
border-top-style, border-right-style, border-bottom-style, border-left-style

border-top-color, border-right-color, border-bottom-color, border-left-color

```
BLOCKQUOTE{
    margin: 1em 3em;  /*上下, 左右*/
    border-width: thin 0em;
    border-style: dashed none;
}
H1{
    border-bottom: thin solid #800;
}
```

5.6.1. マージン・パディングとは

マージン・パディングなどの用語を簡単に説明します（図5-15）.

図5-15. マージン・パディング・テキストインデントの関係

部分	意味
マージン	その要素のボーダーライン（枠線）までの余白，マイナス可
パディング	ボーダーライン（枠線）から文字開始までの余白
テキストインデント	パディング内の行頭のインデント，マイナス可

上マージンをマイナスにすると，直前のブロックと文字が重なり合います．テキストインデントをマイナスにすれば，行頭がパディングよりも左に突出します．ボーダーライン（枠線）を指定しない場合でも，マージンとパディングは別のプロパティです．表示されていないだけでラインそのものは存在すると解釈してください．

5.6.2. 一括指定

※ margin　マージン

値	[< 一般サイズ > | < パーセント > | auto]{1,4} /*auto（自動処理）… 左右マージン：要素の width と左右 padding をひいた残り 上下マージン：不明 */
初期値	0
適用対象	全要素
継承される？	no /* 子要素は，自分のマージンを親要素のマージンに追加する */
パーセントの意味	もっとも近いブロック表示の親要素の width に対する割合 /* 上下マージンでも width に対するパーセント */
マイナス値	可

厳密には，auto は padding などのプロパティの状態により，複雑な計算過程を経て決定されます．しかし，その詳細な解説は仕様書に譲ることにし，本書では簡潔に紹介するにとどめます．

要素に対する余白で，その要素のボーダーライン（枠線）の外側に設定されます．マイナス値も指定可能です．

Lie and Bos(1)は，auto は左右マージンにのみ指定する，と記しています．文法的には上下マージンにも auto が指定できますが，CSS1 および 2 の仕様書のどこにも上下マージンの auto 指定結果について言及はありませんでした．

margin や padding はインライン系要素にも適用できますが，その場合にどう処理されるかは仕様書には書かれていません．

指定の{1,4}は，[　]の中身を1つ，2つ，3つ，4つ書いた場合それぞれの意味が異なることを表わしています．これは，padding，border-width などでも同様です．

回数	意味
1 回	上下左右を同時に指定
2 回	上下，左右の順で指定
3 回	上，左右，下の順で指定
4 回	上，右，下，左の順で指定

```
margin: 1em; /* 上下左右ともに 1em*/
margin: 1em 0em;  /* 上下は 1em, 左右は 0em*/
margin: 1em auto 2em;  /* 上は 1em, 左右は auto, 下は 2em*/
margin: 1em 1em 0em 30%;  /* 上, 右, 下, 左 */
```

マージン指定は継承されませんが，子要素のマージンは親要素のマージンに自分のマージンを追加したものになります．すなわち，子要素にマージンを指定しなければ，親要素と同じマージンになります．

マージン指定のとらえかた

```
UL{ margin: 0em 0em 0em 2em;  /*左2em*/}
```

となっていれば，ULがBODY中に現われようが，BLOCKQUOTE中に現われようが，BODY中のUL中のULとして現われようが，常に直接の親要素のマージンに2emを追加して表示されます．

```
<BODY>
<P>ここがBODYのマージンだとする．</P>
    <UL>
    <LI>「BODY」に対して2em
        <UL>
        <LI>「BODY UL」に対して2em
        </UL>
    <LI>「BODY」に対して2em
    </UL>
</BODY>
```

すなわち，マージンははじめから相対的な位置関係を指定すると考えてください．そのため，指定は継承されないのです．

※ padding　パディング

値	[< 一般サイズ > | < パーセント >]{1,4}
初期値	0
適用対象	全要素
継承される？	no
パーセントの意味	もっとも近いブロック表示の親要素のwidthに対する割合
マイナス値	不可

要素に対する余白で，その要素のボーダーライン（枠線）の内側に設定されます．マイナス値は許されません．

※ border-width　ボーダー幅

値	[thin | medium | thick | < 一般サイズ >]{1,4} /* 細い，普通，太い…具体的なサイズはWWWブラウザに一任 */
初期値	medium
適用対象	全要素
継承される？	no
マイナス値	不可

上下左右のボーダー幅を変えることで，多彩な表現が可能です（**図5-16**）．また，インライン系要素にもボーダーを指定できます（**図5-17**）．

図5-16．ボーダー指定のバリエーション

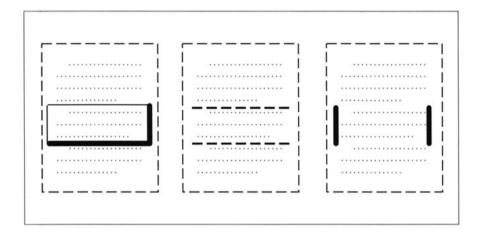
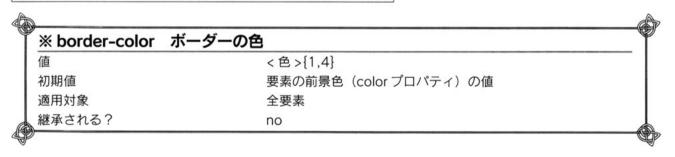
```
{border: thin thick thick thin; border-style: solid;}/*上，右，下，左*/
{border: thin none; border-style: dashed none;}/*上下，左右*/
{border: none 1em; border-style: none solid;}
```

（Lie and Bos(1)p128を参考にしました）

図5-17．ボーダー付きインライン系要素

桃太郎の歌といえば，「ももたろうさん，ももたろさん，お腰に付けたきび団子…」というものですが，これはいつ頃から歌われるようになったのでしょうか。

（折り返し部分にはボーダーが付かない）

※ border-color　ボーダーの色

値	<色>{1,4}
初期値	要素の前景色（colorプロパティ）の値
適用対象	全要素
継承される？	no

ボーダーの色を指定します．

CSS2では，「transparent」（透明）キーワードが値として追加されました．

border-color: url(*.gif)は?

CSS2 までの仕様では，border に画像を割り当てるためのプロパティが用意されていません．実装されるかどうかはともかく，仕様としてはあってもよいと（筆者は）思うのですが…

これも，（W3C の）外部者だから言えることなのでしょうか．

※ border-style　ボーダーの形

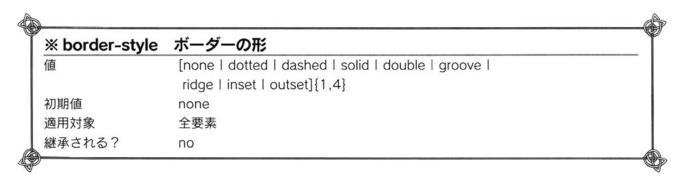
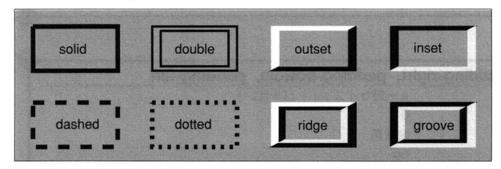
値	[none	dotted	dashed	solid	double	groove	ridge	inset	outset]{1,4}
初期値	none
適用対象	全要素
継承される？	no

ボーダーの形状は，平面的なものから立体的なものまで複数が用意されています（**図 5-18**）．これらも上下左右別々に指定できます（CSS1 仕様書では{1,4}とはされていませんが，単なるミスでしょう．実際，CSS1 仕様書内の記述例でも{1,4}として扱われています）．

図 5-18．border-style

なお，ボーダースタイルの初期値は「none」のため，スタイルを明示しなければボーダーは表示されません．

CSS2 では，TABLE 用に「hidden」というキーワードが追加されました．枠付きのセルに隣接した場合，「none」のセルは隣接セルの枠を表示させますが，「hidden」のセルは隣接セルの枠を表示させません（7 章を参照）．

※ border, border-top, border-right, border-bottom, border-left

値	<border-width> || <border-style> || <border-color>
初期値	個々のプロパティの初期値
適用対象	全要素
継承される？	no

ボーダーの幅，スタイル，色を一括指定するためのプロパティです．borderプロパティは上下左右を一括に，その他は名前が示す部位を担当します．

5.6.3. 単体指定

それぞれ，名前が示す通りの部位を担当します．

※ margin-top, margin-right, margin-bottom, margin-left

値	＜一般サイズ＞ | ＜パーセント＞ | auto
初期値	0
適用対象	全要素
継承される？	no
パーセントの意味	もっとも近いブロック表示の親要素のwidthに対する割合
マイナス値	可

※ padding-top, padding-right, padding-bottom, padding-left

値	＜一般サイズ＞ | ＜パーセント＞
初期値	0
適用対象	全要素
継承される？	no
パーセントの意味	もっとも近いブロック表示の親要素のwidthに対する割合
マイナス値	不可

※border-top-width, border-right-width, border-bottom-width, border-left-width

値	thin | medium | thick | ＜一般サイズ＞
初期値	medium
適用対象	全要素
継承される？	no
マイナス値	不可

5.6.4. CSS2 追加分

※ border-top-color，border-right-color，border-bottom-color，border-left-color

値	< 色 > I transparent
初期値	要素の前景色（color プロパティ）の値
適用対象	全要素
継承される？	no

※ border-top-style，border-right-style，border-bottom-style，border-left-style

値	[none I dotted I dashed I solid I double I groove I ridge I inset I outset I hidden]
初期値	none
適用対象	全要素
継承される？	no

（注：これらがCSS1に用意されていなかったのは，単なるミスだと思われます.）

5-7 表示位置調整

▶ **単体指定**

width, height ・・・・・・・・・・・・・表示の横幅, 縦幅
float ・・・・・・・・・・・・・・・・・・・・・・・フロートと回り込み
clear ・・・・・・・・・・・・・・・・・・・・・回り込みの解除

▶ **一括指定**

なし

▶ **CSS2 追加分**

min-width, min-height・・・・・・横幅・縦幅の最小
max-width, max-height・・・・・横幅・縦幅の最大

これら以外に関しては7章で解説します.

```
H1 IMG{
    width: 1.5em;
}
H1, H2, H3, H4, H5, H6{
    clear: both;
}
```

5.7.1. 単体指定

※ width，height　表示の横幅，縦幅

値	< 一般サイズ > | < パーセント > | auto /*auto（自動処理）… ブロック系要素の場合：width は左右のマージン・パディングをひいた残り，height は表示に必要な幅 置換要素の場合：表示に必要な幅 */
初期値	auto
適用対象	ブロック系要素と置換要素
継承される？	no
パーセントの意味	もっとも近いブロック表示の親要素の width/height に対する割合
マイナス値	不可

autoの処理は，本当はもっと複雑なものですが，その詳細な解説は仕様書に譲ることにし，本書では簡潔に紹介するにとどめます．

表示の横幅・縦幅を指定するプロパティですが，ブロック系要素の場合，特別な効果を狙わない限りはautoとします．

CSS1 仕様書は，ブロック要素の width ・ height を明示したにもかかわらずその中に文字が表示しきれない場合は，部分的にスクロールバーのようなものを提供してもよいが，heightの指定を無視してもよいとしています．CSS2 仕様書では，はみ出した場合の処理方法を指定するためのプロパティが新たに用意されています（7 章を参照）．

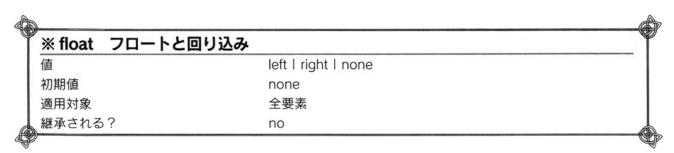
置換要素（画像など）では，指定したサイズに画像が拡大・縮小されます．widthかheightの一方だけを明示し他方をautoとすれば，比率を保ったまま処理されます（**図5-19**）．

図5-19．比率を守った引き延ばし

```
width:4cm;
/*height:auto;*/
```

なお，パーセント指定は「最も近いブロック表示の親要素（the generated box's containing block）…コンテナブロック」（7章を参照）のサイズに対する割合です．自分自身の本来のサイズに対する割合では無いので，ご注意ください（CSS1のheightプロパティには，パーセント指定は許されていません．CSS2では変更されました）．

※ float　フロートと回り込み

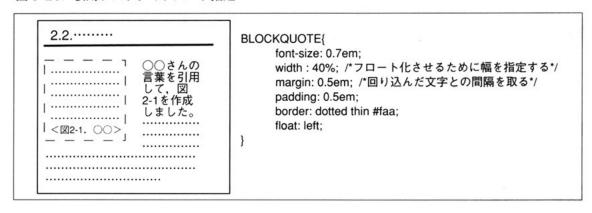
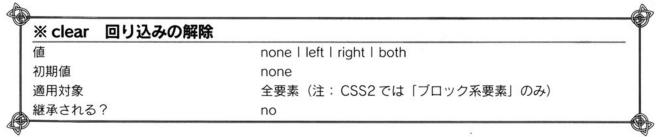
値	left I right I none
初期値	none
適用対象	全要素
継承される？	no

フロートがleft（right）に指定された要素は，displayプロパティの値にかかわらずブロック表示に変更され，その親要素のwidthの左端（右端）に移動すると同時に，その“反対側への”後続テキストの回り込みを許可します．したがって，「float:left」とすると，右側への回り込みが許可されます．

floatプロパティは表示の位置決めのためのプロパティではないので，属性値に「center」はありません．文字の回り込みを伴わない単なる表示位置の調整を行いたい場合は，margin・width・text-alignプロパティをご利用ください．

CSSでは，画像だけでなく文字もフロート化できます．テキストで作った図表や引用ブロックをフロート化させても面白いでしょう（**図5-20**）．

図5-20．引用ブロックのフロート指定

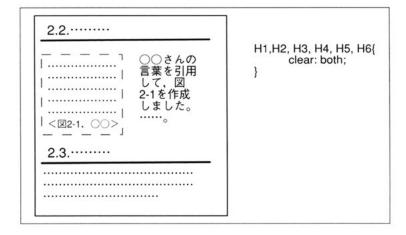
※ clear　回り込みの解除

値	none I left I right I both
初期値	none
適用対象	全要素（注：CSS2では「ブロック系要素」のみ）
継承される？	no

clearプロパティがleft（right）に指定されていると，その要素が現われた所で，float:left（right）によって生まれた回り込みが解除されます．

clearプロパティを用いれば，「新しい見出しが始まったら，回り込みは解除する」などの指

定が可能です（**図 5-21**）．

回り込み解除指定は，HTML3.2 や 4.0-Transitional の BR 要素の CLEAR 属性でも行えます．しかし，BR 要素を埋め込むよりも，見出し要素などに clear プロパティを設定するほうが論理的です．なお，HTML の BR 要素のスタイルシートの CLEAR 属性は，both ではなくて ALL というキーワードを採用していました．混乱の無いよう気を付けてください．

図 5-21．見出しでフロートを解除

5.7.2. CSS2 追加分

※ min-width，min-height　横幅・縦幅の最小値

値	< 一般サイズ > | < パーセント >
初期値	0
適用対象	ブロック系要素と置換要素
継承される？	no
パーセントの意味	もっとも近いブロック表示の親要素の width/height に対する割合
マイナス値	不可

※ max-width，max-height　縦幅・縦幅の最大値

値	< 一般サイズ > | < パーセント > | none
初期値	none
適用対象	ブロック系要素と置換要素
継承される？	no
パーセントの意味	もっとも近いブロック表示の親要素の width/height に対する割合
マイナス値	不可

これらのプロパティが指定されていると，width や height の指定値（あるいは auto による計算結果）がこの範囲の外である場合，こちらの値が採用されます．

5-8 その他の表示調整系

▶ **単体指定**

display …………………表示形式
white-space ……………空白の取り扱い
list-style-type ……………リストマークの種類
list-style-image …………リストマークの画像指定
list-style-position …………リストマークの内側表示・外側表示

▶ **一括指定**

list-style

▶ **CSS2 追加分**

marker-offset ……………マーカー（リストの行頭）の位置調整

```
UL{list-style: circle url(green-mark.gif) }
UL UL{list-style: disc url(red-mark.gif) }
UL UL UL{list-style: square url(yellow-mark.gif) }
```

5.8.1. 単体指定

※ display　表示形式

値	inline	block	list-item	run-in	compact	marker | table	inline-table	table-row-group | table-header-group	table-footer-group	table-row | table-column-group	table-column	table-cell	table-caption | none
初期値	inline（注：CSS1 では block） /*WWW ブラウザ標準スタイルシートによって 　HTML 文書のブロック系要素は block, 　インライン系要素は inline とされていると思われる */
適用対象	全要素
継承される？	no

display プロパティは，各要素の表示のされかたを指定します（**表 5-7**）．また，display プロパティの指定を元に，各種プロパティが適用されるか否かが判断されます（5.1 節を参照）（変わるのは表示上の解釈だけで，マークアップにおける性質は変わりません）．

表 5-7．display プロパティの意味

指定	結果
block	ブロック（その要素の開始・終了で改行する）
inline	インライン（その要素の開始・終了では改行しない）
list-item	要素の行頭に「マーカー」が付けられる特別な block
run-in	後述する特殊な block
compact	後述する特殊な list-item
marker	後述する特殊な区分で，block でも inline でも無い
table 系	テーブルに相当する特殊な区分（7 章を参照）
none	表示しない

（CSS1 では，block，inline，list-item，none のみ）

通常は，HTML の本来の位置づけとは異なる効果を得たい場合にのみ指定します．たとえば，UL 要素のリスト項目を横に並べたい場合は，その内部の LI 要素を inline として指定します（**図 5-22**）．

図 5-22. インライン指定の LI 要素

```
<STYLE TYPE="text/css">
    UL LI{ display: inline}
</STYLE>

<UL>
    <LI>1st,
    <LI>2nd.
</UL>
```

```
1st, 2nd.
```

「display: compact」と指定された要素は，その直後のブロックを自分の横にならべても表示幅からはみ出さない場合に，ブロックを横にならべるように機能します（**図 5-23.** compact 指定）．すなわち，表示幅からはみ出さない場合は，自分と直後のブロックをつなげてひとつの block であるかのように振る舞うことになります．

図 5-23. compact の適用例

```
UL.NAME{display: compact}
<UL CLASS="NAME BEATLES MENBER">
    <LI>John
    <LI>Paul
    <LI>Ringo
    <LI>George
</UL>
```

```
|―――――――――――――――|表示幅
  ・John  ・Paul  ・Ringo
  ・George
```

「display:run-in」は，おもに見出しに使われる区分で，常にその直後のブロックと自分自身をひとつのブロックであるかのように振る舞わせます．

compact でも run-in でも同じですが，要素の表示がつながってしまうため，:after 疑似要素と content プロパティを有効に利用するべきでしょう（**図 5-24**，7 章を参照）．

図 5-24. run-in と:after 疑似要素の適用例

```
H3{display: run-in;}
H3:after{content:"："}

<H1>日記</H1>
<H2>1998年5月21日</H2>
<H3>天気</H3>
<P>晴れ．湿気が無いからまだいいけど，かなり暑い．</P>
<H3>覚え書き</H3>
<P>あいつに3回も電話したのに，一度もつながらなかった．</P>
```

```
|―――――――――――――――――――――――――――――――――――|表示幅
日記
1998年5月21日
天気：晴れ．湿気が無いからまだいいけど，かなり暑い．
覚え書き：あいつに3回も電話したのに，一度もつながらな
かった．
```

なお，仕様書によると，直後のブロックがfloat指定あるいは絶対位置指定（7章を参照）されている場合は，compactやrun-inは機能しません．

※ white-space　空白の取り扱い

値	normal | pre | nowrap /* 標準 | プレ整形 | 自動折り返しを拒否 */
初期値	normal /*WWWブラウザ標準のスタイルシートによって 　PRE要素はpre指定されていると思われる．*/
適用対象	ブロック系要素
継承される？	yes

ソース中の空白・タブ・改行をどのように表示するかを指定します（**表5-8**）．

表 5-8. white-space プロパティの意味

指定	連続した空白・タブ・改行	表示幅に合わせた自動折り返し
normal	一つの空白として表示	する
pre	ソースのまま表示	しない（ソース中の“改行”で改行）
nowrap	一つの空白として表示	しない（BR要素でのみ改行）

※ list-style-type リストマークの種類

値	disc ǀ circle ǀ square ǀ decimal ǀ lower-roman ǀ upper-roman ǀ lower-alpha ǀ upper-alpha ǀ none /* ● ｜ ○ ｜ ■ ｜ 1,2,3 … ｜ i,ii,iii … ｜ I,II,III … ｜ a,b,c … ｜ A,B,C … ｜ 無し */ /* これはCSS1の仕様である */
初期値	disc
適用対象	list-item 要素
継承される？	yes

list-item 要素の行頭マーカーの種類を指定します．ただし，list-style-image プロパティが有効の場合は，そちらが優先されます．

CSS2 では，list-style-type の属性値が大幅に追加されました（**表 5-9**）.

表 5-9．CSS2 の list-type-style

値	説明
decimal-leading-zero	頭にゼロを付けた数字（01，02，…）
lower-latin，upper-latin	alpha と同じ（a，b，c …）
lower-greek	ギリシャ文字（α，β，…）(alpha，beta，…)
hebrew	伝統的なヘブライ式
georgian	ジョージア式（an，ban，gan …，he，tan，in，in-an …）
armenian	伝統的なアルメニア式
cjk-ideographic	象形文字
hiragana	あ，い，う…（a，i，u …）
katakana	ア，イ，ウ…（A，I，U …）
hiragana-iroha	い，ろ，は…（I，ro，ha …）
katakana-iroha	イ，ロ，ハ…（I，RO，HA …）

goergian の例は，CSS2 仕様書に書かれていたものです．筆者の不明で，goergian の詳細は調べられませんでした．イギリスのジョージア王朝のことでしょうか?

入れ子リストにおけるマーカー指定での注意

指定は継承されるため，リストの入れ子レベルごとにマーカーを変更したい場合は，すべての入れ子レベルについて明示的に指定する必要があります（図5-25）．

図5-25．入れ子指定を怠ると

```
UL{list-style-type: disc}
UL UL{list-style-type: circle}
●入れ子1
    ○入れ子2
    ○入れ子2
        ○入れ子3（入れ子2の指定を継承）
        ○入れ子3
●入れ子1
```

WWWブラウザ標準スタイルシートの記述に期待するという手もあるのですが，IE4.0もNetscape Navigator4.0も，「子孫（下降）セレクタ」を優先するようです．なお，CSS2であれば，「子セレクタ」を上手く利用することでこの不具合を避けられます．

数え上げリストの数値指定

HTML3.2およびHTML4.0Transitionalでは，OL要素のSTART属性で数え上げリストの数字をコントロールできました．しかし，HTML4.0Strictではこの属性は削除されました．さて，その代わりに用意されたCSSのプロパティは何でしょう？

実は，CSS1では相当するプロパティは用意されませんでした．しかし，CSS2ではカウンタという仕組みが導入され，「カウントを開始する数字」「一度にカウントアップする量」などを自在に指定できます．そのかわり，数字を制御するには，複数のプロパティを組み合わさなければいけません．

詳しくは7章を参照してください．

※ list-style-image　リストマークの画像指定

値	<url> | none
初期値	none
適用対象	list-item要素
継承される？	yes

リスト行頭のマークを任意の画像で指定します（**図5-26**）．画像を表示できない場合は，list-style-typeプロパティの指定が有効になります．

図 5-26. リストマークを画像に

```
UL{ list-style-image: url(star.gif) }
```

※ list-style-position　リストマークの内側表示・外側表示

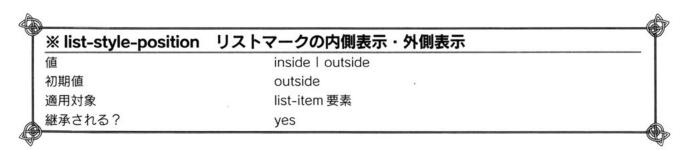
値	inside | outside
初期値	outside
適用対象	list-item 要素
継承される？	yes

マークが要素ブロックの内部に表示されるのか外部に表示されるのかを指定します．この違いは文章が折り返し表示されたときに明確になります（**図 5-27**）.

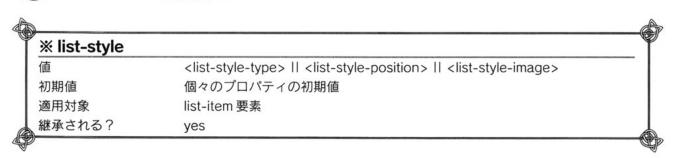
図 5-27. list-style-position プロパティの意味

inside：

outside：

5.8.2. 一括指定

※ list-style

値	<list-style-type> || <list-style-position> || <list-style-image>
初期値	個々のプロパティの初期値
適用対象	list-item 要素
継承される？	yes

5.8.3. CSS2追加分

マーカーとは

リストの行頭マーカーの振る舞いは「フロート指定したもの」に似ていますが，必ずしも同一ではありません．そこで，CSS2では新たに「マーカー」という概念を定義しました．

CSS2では，「マーカー」の実体を，「display: marker」と指定された特殊な「:before」疑似要素として処理します．list-style系のプロパティを指定するのは「list-item」系要素ですが，実際に調整されるのはリスト項目行頭の「マーカー」（疑似）要素なのです．

マーカーは元の要素とは独立した要素だと考えますので，たとえば，LI要素に背景画像を指定しても，マーカーには適用されません．マーカーのサイズや色，背景などの調整は，端的にはLI:before疑似要素に指定することになります．

なお，任意の要素の「:befor」（あるいは:after）疑似要素に「display: marker」を指定することによって，その要素の前後にマーカーを表示させられます．しかし，リスト以外では利用すべきではないでしょう．

※ marker-offset	マーカーの位置調整
値	< 一般サイズ > | auto
初期値	auto
適用対象	マーカー
継承される？	no
マイナス値	可 /* 正であれば，マーカーがより離れる・負であれば，近づく */

マーカーにはpaddingとborderは指定できますが，marginは指定できないことになりました．そのかわりに，marker-offsetプロパティを利用します．

marker-offsetは，マーカーと元要素との間隔を調整するためのプロパティです．マーカーの性質上，水平方向の調整のみが可能です．

【引用文献】

(1) Håkon Wium Lie and Bert Bos（1997）Cascading Style Sheets: addison wesley, 279pp

W3CのCSS仕様調整担当者であるHåkon Wium LieとBert Bosが記述したCSS1の仕様解説書です．ただし，HTML4.0草案検討中の段階で出版されたために，HTML文書との連携の仕様が最終的に採用されたものと異なります．

road to mastering CSS

６章では，実際にスタイルシートを構築する上で重要になる概念や，無理のないシートにするための「考え方」を紹介します．

逆説的な導入

6.1.1. CSS1のみによる影付き文字

CSS2では「影付き文字」のための専用プロパティが用意されましたが，Internet Explorer4.0もNetscape Navigator4.0もまだこれをサポートしていません．しかし，**CSS1の機能のみ**でも，マイナスの上マージンとテキストインデントを利用すれば，「影付き文字」効果を簡単に作れるのです（**図6-1**）．

図6-1．影付き文字

文字に影を付ける

少しずらして文字を描写すれば，影がついたような感じになります。

「おう，これは簡単で，しかもカッコイイ！ これでこそスタイルシートだ!」．

しかし，本当にそうでしょうか．

図6-1のソースコード

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>text with shadow</TITLE>
<STYLE TYPE="text/css">
    BODY{background: white;}
    H1{
        font: 700 2em sans-serif;
        line-height:1;
    }
    H1 SPAN{
        margin: 0em;
```

```
        display: block;
    }
    H1 SPAN.BEHIND{
        color: gray;
    }
    H1 SPAN.FORWARD{
        color: black;
        margin-top: -1.2em;  /*ここがポイントです*/
        text-indent: -0.2em;  /*ここがポイントです*/
    }
</STYLE>
</HEAD>
<BODY>
<H1>
<SPAN CLASS="BEHIND">文字に影を付ける</SPAN>
<SPAN CLASS="FORWARD">文字に影を付ける</SPAN>
</H1>
<P>少しずらして文字を描写すれば，影がついたような感じになります．</P>
</BODY>
</HTML>
```

6.1.2. 画像にしなかった利点はあるのか

影付き文字などの効果は，ドロー系の画像作成ソフトウェアを用いればより簡単に，しかも高度に作成できます．また，この結果をペイント系レタッチソフトで加工すれば，水でぬらしたようににじませたり，グラデーション，エンボスなどさまざまな効果を実現できます．それをせずに，わざわざスタイルシートを用いた理由は何でしょう．

「だって，はじめに“画像は望ましくない”って言ったじゃないか!」

確かに，画像にしてしまうとデータサイズが大きくなりますし，また画像を表示できない人に情報を伝達できません．しかし，それでも画像でしか得られない効果が必要なときは，やはり画像を用いるべきです．スタイルシートは画像の代替になるものでは**ありません**し，そのように利用するのは**望ましくありません**．

図6-1のHTML文書を，スタイルシートをオフにして表示するとどうなるでしょうか．ソースを見て分かるとおり，

「文字に影を付ける文字に影を付ける」

となります．この文書の見出しは「文字に影を付ける文字に影を付ける」でいいのでしょうか？

この問題は，CSS2のサポートを待てば解決するわけではありません．なぜなら，根本にあるのは，**「本来スタイルシートで行うべきでないことを強行した」**という問題なのです．

- スタイルシートが無いと内容を伝達できないHTML文書になってしまった
- スタイルシートのために文章内容を改ざんしてしまった

このような事態は，1章で挙げたスタイルシートの利点に反するものです．どちらかといえば，「怪しいテクニック」の欠点に近いものです．これは望ましい使い方ではありません．

それでは，このような場合にはどうするべきなのでしょうか？　本来の見出しは「文字に影を付ける」のはずです．そうであれば，HTML文書インスタンスには，

```
<H1>文字に影を付ける</H1>
```

とのみ書かれているべきです．影付き文字として表示したければ，それを画像にし，

```
<H1><IMG SRC="title.gif" ALT="文字に影を付ける"></H1>
```

としておくのが良いでしょう．これならば，画像を表示しない人にも内容を伝達できます．

もちろん，CSS2のtext-shadowプロパティを用いているのはかまいません．

6.1.3. 好ましい「考え方」

スタイルシートはHTML文書の表示結果を調整するものです．しかし，スタイルシートが無いと読めないようなHTML文書を書いてはいけません．では，スタイルシートはどんな形でHTML文書と連携するべきなのでしょう？　筆者の用意した答えはこうです．

HTML文書の表現を助ける目的で，その構成に合わせてスタイルシートを記述しましょう．

2・3章では，HTML文書で構成を明示することを解説しました．その指針にしたがっている限り，作成したHTML文書はスタイルシート無しでも十分文章内容をアピールできています．スタイルシートは本来，その文章アピールを補助的にサポートするものであるべきです．「見出しは見出しらしく」「本文は本文らしく」をモットーにスタイルシートを記述していきましょう（補助的といっても，見栄えに関してはHTML文書を支配するものです）．

以降では，スタイルシートを記述するための考え方を，「全体的な調整」「見出しの調整」の順に具体的に紹介します．最後には，使いまわしの効くスタイルシートについて紹介します．

6-2 全体的な調整

6.2.1.「一般 → 例外」という原則を守る

文書のスタイル表現に限らず，多くの表現には**「一般」**と**「例外」**が存在します．通常，本文中のほとんどの文は同じスタイルで書かれ，その中の特別な語句，すなわち，見出しや強調語句だけが特別なスタイルで書かれています．

CSSの継承（inheritance）という仕組みを利用すれば，この「一般」と「例外」を的確に表現できます．すなわち，BODY要素に対して一般的なスタイルを指定し，各要素には基本的にその指定を継承させます．そして，例外的に指定したい部分を後から付け足していくのが望ましいでしょう（この考え方は，CSS2の仕様書にも明示されています）．

・子要素に共通する一般性質は，親要素に「持ち上げ」よう

6.2.2. BODY要素（一般本文）への指定

基本的に，本文は「読みやすい」ことが大事です．したがって，太さは太すぎず細すぎず，サイズは大きすぎず小さすぎず，フォントファミリーは奇抜でなく…と，**「無難」**なものを目指すのが普通だと思われます．一般的には，大体次のような指定になると思います．

```
/* whitebase.css */ /*原始段階*/
BODY{
    font: medium serif;  /*weight-400: style-nomal: variant-nomal*/
    text-align: left;
    background: #fff;  /*white*/
    color:#000;  /*black*/
    line-height: 1.4;  /*一般に，読みやすいのは1.2～1.7*/
}
/*以降の指定はBODYへのものではありませんが，ここにあげておきます．*/
A:link{color:#63f}
A:visited{color:#f0a}
A:active{color:#f00;}
```

特別に意識すべき点は次のとおりです.

・文字色と背景色はセットで指定する（カスケーディング処理によって「読めない色の組み合わせ」になるのを避けるため）
・行幅の継承の違いを意識し，emの指定と「掛ける値」の指定を使い分ける

mediumか，12ptか？

whitebase.cssでは，BODYのfont-sizeをmediumとしました．これには理由があります．

もし，これをfont-size:12ptとした場合，読者はこのシートによる表示サイズを自分の環境に合わせて変更できません．一方，mediumの具体的なサイズは，読者が**WWWブラウザの設定を変えることで**，読者自身の手で調整できるのです．

ディスプレイの大きさによっては，12ptは小さすぎるかも大きすぎるかもしれません．したがって，筆者はmediumの指定をお勧めします．

6.2.3. インライン系要素への指定

STRONG要素やA（アンカー）要素といったインライン系要素は，本文の中に埋め込まれる特別な語句に相当します．これらが普通の本文と明確に区別できるのは良いことですが，あまり極端にスタイルを変更しては，かえって本文が読みにくくなります（図6-2）．悪い例としては，次のようなものが挙げられます．

・STRONG{font-size: 3em}・・・・・・・・・・大きすぎるため，文の流れが悪くなる
・STRONG{border: 1em groove}・・・・・ボーダーが太すぎる

図6-2. STRONG要素への指定の悪い例

私の誕生日は2月24日です。ひろのみやさんの翌日です。弟の誕生日は4月28日で，こちらはひろひと天皇の前日。
しかも，“ににんがし”と“しにがはち”です。嘘っぽいけど，本当の誕生日です。

これ以外にもいろいろなものが考えられますが，大事なのは**「無理の無い程度に変える」**ように心がけることです．したがって，ボーダーラインや背景画像などの強烈なものは避けることにし，**文字そのものに関する**ちょっとした調整に止めるほうが良いと思われます．次に挙げるものうち，いくつかを組み合わせて指定するとよいでしょう．

- ちょっと文字を太くする，大きくする，フォントファミリーを変える
- ちょっと色を変える，背景色を付ける
- 斜体，太字，点滅，アンダーラインなど
- 単語間隔や文字間隔を詰める，広げる

「WWWブラウザ標準のスタイルシートに期待して，まったく変更しない」のも望ましい態度です．そうすれば，読者がもっとも慣れ親しんだ表示を取り込むことになるのですから．

6.6.1で例に取り上げる「ks.css」では，STRONGに**図6-3**の指定を採用しています．

```
STRONG{
    color:black;
    background: #faa;
    font-weight: bolder;
}
```

図6-3.「ks.css」のSTRONG

/ニセのDTDはいやーん/

本論

最近よくみかけるニセのDOCYTPE宣言で，こういうのがある。

<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">

W3CはそんなDTD（文書型定義）は公開していません。"-//W3C//DTD HTML VERSION（3.2とか4.0とか）//EN"が公開されているDTDです。（なお，それぞれにサブバージョンがありますので，念の為。）「DOCTYPE宣言」とは，「対応するDTD（文書型定義）を守っています」という宣言であるので，存在しないDTDを指すDOCTYPE宣言は「偽札」みたいなものだ（??）。

6.2.4. ブロック系要素への指定

ふつうに考えると，「本文」と「見出し」は別のものです．ところが，HTML4.0ではBODY要素が「本文」であり，その中に「見出し」や「段落」などが「ブロック要素」とひとくくりにされています．

しかし「見出し」は，ほかのブロック系要素とは異なる特別な役割を持っています．すなわち，「見出し」とは本文の流れの変化を主張するための「道しるべ」であり，それに続く内容を端的に表している「短いまとめの言葉」です．したがって「見出し」は，ひと目で本文と区別できるように目立っているべきですし，少々派手すぎても内容把握には支障ありません．つまり「見出し」に関しては，どんどんスタイル変更してもよいでしょう．そこで，「見出し」に関しては，独立して6.4節として詳細に取り上げたいと思います．

ほかのブロック系要素は，「本文」を構成するものです．P要素を「ごく一般の本文」ととらえるならば，BLOCKQUOTE要素などは「特殊な本文」だといえるでしょう．

なかでも，BLOCKQUOTE要素は位置づけのはっきりした要素です．その内容は「ほかの書物などからの引用」なのですから，しっかりと「ごく一般の本文」と流れが区別できるべきでしょう．表現するべきは「流れの変化」ですから，文字そのものはあまりいじらずに，ボーダーラインやマージンなど**「表示の枠」を中心に調整**したほうが良いと思われます（ブロック系要素で文字そのものを派手にいじってしまうと，その内部のインライン系要素の表現に困ることになるでしょう）．そのためには，次のような指定が考えられます．

- マージンを増やす
- ボーダーラインを付ける
- ちょっと文字を小さくする，フォントファミリーを変える
- 斜体にする
- ちょっと色を変える，背景色や背景画像を付ける

6.6.2で例に取り上げる「prince.css」では「マージンを均等に取り，ボーダーを左右のみに」指定し（**図6-4**），「zappa.css」では「左マージンを大きく取り，さらにボーダーも左だけに」指定しています（**図6-5**）．

図6-4.「prince.css」のBLOCKQUOTE

```
BLOCKQUOTE{
    margin: 2em;
    font: italic 1em monospace;
    color: #bbb;
    border-width: 0em thick;
    border-style: solid;
    border-color: #a40;
    padding: 1em 0.5em;;
}
```

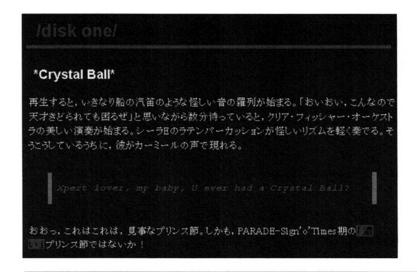



図6-5.「zappa.css」のBLOCKQUOTE

```
BLOCKQUOTE{
    margin:1.5em 1em 1.5em 3em;
    border-width: 0em 0em 0em thick;
    border-style: none none none solid;
    border-color: #969;
    padding: 1em 1em 1em 2em;
}
```

Just Another Band From L.A.

March 1972

全体にどことなくコミックバンドみたいな演奏です。歌詞も。

> 「お山のビリーくんが奥さんと休暇にNYへ遊びに行ったら，なにぶん山だもの，各地に地震やらのひどい被害を起こしてしまった。」
>
> 「これを見たアメリカ政府，ビリーくんに徴兵の赤紙を出しました。果たしてビリーくんはどうなってしまうのか！」

というお話の[Billy the mountain]は，刻々と変化するバック演奏の上でなされる演劇おしゃべりと，ところどころで挿入されるロック，ドゥーワップ，オペラといった歌によって構成された大作。きちんとメロがついている部分のクラシカル／プログレな雰囲気には，「さすがだ」と感心することしきり。(でも，演奏がおもちゃっぽいのはなぜだろう？)

この考え方は，ほかのブロック系要素でも通用するでしょう．

リストの表示に関する指定は，Netscape NavigatorやInternet Explorerなどの標準の指定（入れ子ごとに左マージンが深くなる）があまりにも絶妙な効果を上げています．われわれは，マージンのサイズを変える程度の変更にとどまるべきでしょう．

6-3 emユニットの活用（相対指定のススメ）

ここで話の毛色が変わりますが，実際に「全体的な」スタイルシートの構築に入る前に，emユニットに代表される**相対指定の重要性**に関して説明します．

emユニットを無視すると，とんでもない混乱に陥ることがあります．まずは，emユニットを使わないとどんな難点に突き当たるのかを例示しましょう．

6.3.1. text-indentにおける混乱

たとえば，フォントサイズ10ptの要素に対し，1文字分のインデントを期待するという意味で「{text-indent: 10pt}」と指定したとしましょう．これでは，フォントサイズを変更するたびにインデントの指定も変更しなくては「一文字文のインデント」をキープできません．「{text-indent: 1em}」としていれば，難なく「一文字分のインデント」を実現できるのに…！

```
BODY{
    font-size:10pt;
    text-indent: 1em;
}
/*text-indent: 10ptでも一文字分だが，フォントサイズを20ptに変更したら，
10ptは一文字分では無くなってしまう．一方，1emは常に一文字分である．*/
```

6.3.2. 親子間のfont-sizeにおける混乱

また，BODYのfont-sizeが12ptのとき，H1をその2倍にしたい場合はどのように指定するべきでしょうか？　たいていの場合H1の親要素はBODYなので，「H1{font-size: 2em}」でOKです（注：関係を明示するために，「BODY H1{font-size: 2em}」と文脈付きで記述しても構いません）．これを「H1{font-size: 24pt}」などとしてしまうと，BODYのfont-sizeを10ptに変更したときに，あわせてH1の指定まで変更する羽目に陥ります．

6.3.3. 問題の本質と解決の指針

以上で紹介したのは，実は「**バランス**をキープする上での問題」です．

スタイルシートは個々の要素のスタイルを別々に指定していくものですが，指定が個々で完結しているわけではありません．すなわち，スタイルシートは，各部分の**相対的なバランス**を考慮しながらくみ上げていくものであり，そのバランスをキープすることが重要課題になるのです．

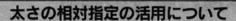
ところが，各種指定を「絶対値」で指定してしまうと，ある部分を変更したくなった場合，連鎖して（バランスを保つために）別の部分を変更しなければならなくなります．この際にどこか1つでも連鎖変更を怠れば，思わぬところで意図した表示結果ではないものが表示されてしまいます．シートが複雑になればなるほど，「どこを直せばいいのかわからなくなる」ことは必至です．

この混乱を避けるには，「**はじめから相対指定で**シートをくみ上げていくべき」です．相対指定であれば，ある1個所のサイズを変更したとしても，バランスをキープするためにシート全体を書き直す必要はありません．なぜなら，はじめから「バランス」だけが書かれているのですから．

相対指定のためのサイズ単位としてもっとも柔軟性に富んでいるのがemユニットです（emユニットは，font-sizeプロパティにおいては親要素との相対関係を，その他のプロパティにおいてはその要素のfont-sizeとの相対関係を表現します）．emユニットを用いれば，シートの中身が「何文字分のインデントやマージン」というわかりやすい表現になります．シートをわかりやすく記述し，しかも相対性を確保するために，存分にemユニットを活用するべきです

プロパティによっては，パーセント指定が可能なものもあります．パーセントもemユニットと同じように有用です．

しかし，すべてが「相対」では何も表示できません．すなわち，どこかに1つ，基準となる絶対値が必要です．そのためには，**基準絶対値は「BODYに指定するfont-size」で指定する**のがよいでしょう．すなわち，BODYのfont-sizeのみを絶対サイズで記述し，残りのものはすべてemユニットやパーセント指定などの相対指定にするわけです．そうすれば，BODYのfont-sizeを変更しただけで，ほかの部分の表示もバランスを保ったまま（自動的に連鎖して）変更されます．

太さの相対指定の活用について

font-sizeには，larger・smallerといった相対指定専用のキーワードが用意されています．しかし，一段階の「大きめ」「小さめ」しか指定できず，emユニットに比べると，表現力が貧弱です．

同様に，太さ（font-weight）にもbolder・lighterといった相対指定専用のキーワードがあります．しかし，やはり一段階の「太め」「細め」しか指定できず，表現力が貧弱です．といっても，emユニットでは太さは指定できないため，相対指定に関してはもう選択肢がありません．そこで，筆者は太さに関してはすべて絶対指定を採用しています（先に述べた理念と矛盾していますが，暫定的手段です）．

bolder・lighterの指定は，インライン系要素への指定には有用だと思います．たとえば，STRONG要素の表示がH1要素中に現れようとP要素中に現れようと，確実に1段階太く・細く表示させることができるからです．

しかし，ブロック系要素の親子関係では，「1段階」の変化よりも急激に変化するほうが望ましい場合が多いでしょう．たとえば，H1要素はBODY要素よりも4段階以上太くなって欲しいと思っても不思議はないでしょう．

このような場合にはfont-sizeにおけるemユニットのようなものが欲しいのですが，現存のスタイルシートの仕様ではそのような単位が用意されていません．そこで，本書はあえて絶対指定で太さを指定しました．

6-4 見出しの調整

6.4.1. サイズと左マージン

既に述べたとおり，見出しは**本文よりも目立っているべきだと**考えられます．

- ずば抜けて太く，大きくする
- フォントファミリーを変える
- 右寄せ，真ん中寄せを用いる

見出しが長く複数行にわたった場合でも“1つの見出し”だと強調するために，行幅は1.0～1.2程度にしておくとよいでしょう．

さらに，見出しレベル間に違いを持たすことが重要です．レベルの上下が自然に読み取れるよう工夫しておくとより望ましいでしょう．一般に，見出しレベルの変化は，サイズの大小や左マージン量の増減で示されます．

- サイズ大 ……………………レベルが高い
- 左寄せ時に左マージン多……・レベルが低い

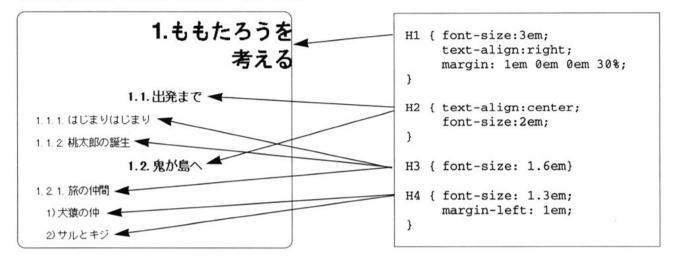
図6-6に見出しを調整したスタイルシート例を提示します．ここでは，どのレベルの見出しにも共通する指定を先に記述しておくと，シート全体が把握しやすくなることをご確認ください．これも「一般→例外」という考え方のひとつです．

```
/* atomic.css */
@import url(whitebase.css);

/* 見出し全般への指定 */
H1, H2, H3, H4, H5, H6{
    font-weight: 700;
    font-family: sans-serif;
    line-height: 1.1;
}/* 以上のスタイルシートは，この章を通じてベースとして利用します．*/
```

```
/* standard.css */
@import url(atomic.css);
H1{
    font-weight:900; /*更に太く*/
    font-size:3em;  /*ずば抜けて大きく*/
    text-align:right;  /*右寄せで変化を強調*/
    margin: 1em 0em 0em 30%;
        /*“右寄十左に大きなマージン”は，文字折り返し時に効果が出る*/
}
H2{
    text-align:center;  /*真ん中よせで変化を強調*/
    font-size:2em;  /*第1レベルよりは小さく，3，4よりは大きく*/
}
H3{    font-size: 1.6em}
H4{
    font-size: 1.3em;
    margin-left: 1em; /*レベル3と4の違いは，左マージンで*/
}
H5{
    font-size: 1.2em;
    margin-left: 1.3em;
}
H6{
    font-size: 1.1em;
    margin-left: 1.5em;
}
}
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>ももたろうを考える</TITLE>
<LINK REL="STYELSHEET" TYPE="text/css" HREF="whitebase.css">
<LINK REL="STYELSHEET" TYPE="text/css" HREF="atomic.css">
<LINK REL="STYELSHEET" TYPE="text/css" HREF="standard.css">
</HEAD>
<BODY>
<H1>1.ももたろうを考える</H1>
    <H2>1.1.出発まで</H2>
        <H3>1.1.1.はじまりはじまり</H3>
        <H3>1.1.2.桃太郎の誕生</H3>
    <H2>1.2.鬼が島へ</H2>
```

```
          <H3>1.2.1.旅の仲間</H3>
               <H4>1)犬猿の仲</h4>
               <H4>2)サルとキジ</H4>
</BODY>
```

図 6-6. 見出しレベルをサイズと左マージンで表現

@import と LINK 要素を併記

standard.css には @import で atomic.css が取り込まれています．atomic.css には whitebase.css が取り込まれています．したがって，LINK 要素で standard.css を指定しただけで，この 3 つのシートを指定したのと同じ効果が得られます．

しかし，この例では**わざわざ** 3 つのシートへの LINK 要素がかかれています．なぜでしょうか？

実は，Netscape Navigator4.0 が @import 構文をサポートしていないことへの配慮です．作業の手間は増えますが，過渡期にはこのようなテクニックが必要になるものです．

スタイルシートの分割と再利用

どのスタイルシートでも共通に使いうる土台的な部分は，独立したシートとして保存しておき，各シートから @import して利用すると便利でしょう．また，コンポーネント化の発想に基づいて，見出しだけ・段落だけ・引用ブロックだけのシートを複数用意し，LINK 要素でそれらを組み合わせながら新たな効果を編み出していくのも楽しいと思われます．

しかし，筆者は「この場合はシートを分離しておくべきだ」という絶対的な方針をお伝えすることはできません．なぜなら，趣味や目指す指定によって，分離ルールは大きく変化するものだからです．

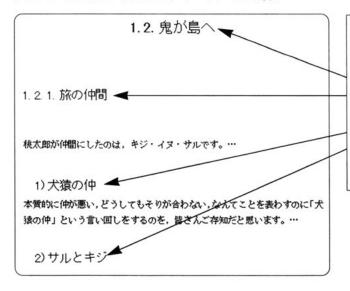
この章におけるシートの分離・コンポーネント化はあくまでも筆者のお勧めですが，もしこれが参考になれば幸いです．

6.4.2. 上下マージン

見出しは，**ざっと眺めただけでもすぐに発見できるほど**目立っているべきです．効果的に見出しをアピールするには，見出しの上下に大き目のマージンを入れるのが良いでしょう．

- 上下マージン大…見出しと本文が遠い…内容の大きな変化をアピール…見出しレベルが高い
- 上下マージン小…見出しと本文が近い…内容の小さな変化をアピール…見出しレベルが低い

ただし，見出しの“上”は前セクションとの境，“下”はその見出しが表わすセクションとの接点です．したがって，上マージンは大きめに，下マージンは小さめにするほうが望ましいでしょう（**図6-7**）．

図6-7. 見出しと本文を上下マージンで差別化

```
/*standard2.css*/
/*上下マージンを追加*/
@import url(standard.css);
H2{margin: 2.5em 0em}  /*上下マージン2.5em*/
H3{margin: 3em 0em 2em 0em}
/*上, 右, 下, 左*/
H4{margin: 2.5em 0em 0em 1em}
H5{margin: 2em 0em 0em 1.3em}
H6{margin: 2em 0em 0em 1.5em}
```

6.4.3. ボーダーライン

見出し上下にボーダーラインを引けば，もっと直接的にセクションの区切りを表現できます．全てのレベルの見出しにボーダーがあってはくどすぎますが，ボーダー付きの見出しを適度に混ぜることは効果的です．

- 見出しの上のボーダー …前セクションの終わりを強調
- 見出しの下ボーダー ……次セクションの始まりを強調

border-style には立体的なものが幾つか用意されていますが，平坦な solid にした方が文字と馴染みやすいと思います．

図6-8. 話の転換をアピールするボーダー

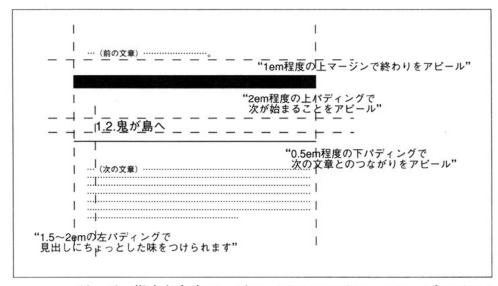
```
H2{
    border-width: 1em 0em thin 0em;  /*上, 右, 下, 左*/
    border-style: solid none:  /*上下, 左右*/
    padding: 2em 0em 0.5em 1.5em:
}
```

ボーダー指定と左右マージン・パディングやtext-alignプロパティを適切に組み合わせると，見栄えの「感じ」はかなり変わります．見出しレベルの違いを考慮しながら，いくらか調整してみるとよいでしょう（**図6-9**）．

図6-9. 少々凝ったボーダー例

```
/*「図6-5. 少々凝ったボーダー例」のスタイルシート*/
/* fancy.css */
@import url(standard2.css);
BODY{
    background: #ffe; /*淡い黄色～クリーム色*/
}
H1{
    margin: 1em 0em 2em 20%;  /*standard.cssより左マージンを減らした*/
}
H2{
    text-align: left;  /* standard.cssでcenterにしたのを戻した*/
    font-weight: 900;  /*装飾に負けないように，太く*/
    border-width: thick 0em 0em 1.5em;  /*上と左に太さの違うボーダーを*/
    border-style: solid none none solid;
    border-color:#390;
    padding: 0.5em 0.5em 0em 0.5em;
}
```

次のように手っ取り早く指定しないのは何故でしょう？

```
border-top: thick solid #390;
border-left: 1.5em solid #390;
```

実は，これらのプロパティをNetscape Navigator4.0がサポートしていないのです．
現在のサポート状況に関しては，巻末資料をご覧ください．

```
H3{
    margin: 2.5em 1em 1.5em 0.5em;  /*右に隙間…ボーダーが左に寄る*/
    border-width: thin 0em 0em 0em;  /*上だけに細いボーダーを*/
    border-style: solid none none none;
    padding: 0.5em 0em 0em 0.5em;
}
```

6.4.4. 通し番号を装飾（spanning）

(1) <SPAN CLASS="NUMBER"> ～ </SPAN> マークアップ

見出しを更に装飾するために，通し番号（1.1.や1.3.2.）と見出し語句そのものを別のものとして扱ってみましょう．

そのためには，スタイル指定に先立って，HTML文書中にマークアップで「通し番号」要素と「見出し語句」要素を明示する必要があります．これらはHTML本来の要素としては用意されていないため，SPAN要素とCLASS属性を用いてマークアップします．

```
<H1>
<SPAN CLASS="NUMBER">1.</SPAN>
<SPAN CLASS="MAIN">ももたろうを考える</SPAN>
</H1>
    <HR>
    <H2>
    <SPAN CLASS="NUMBER">1.1.</SPAN>
    <SPAN CLASS="MAIN">出発まで</SPAN>
    </H2>
    <HR>
        <H3>
        <SPAN CLASS="NUMBER">1.1.1.</SPAN>
        <SPAN CLASS="MAIN">はじまりはじまり</SPAN>
        </H3>
```

(2) インラインのまま装飾

単純にSPAN.NUMBERのサイズと色を変えただけでも，見栄えにメリハリが生まれます(図6-10)．

図6-10. 通し番号をインラインのまま装飾

```
/*fancy2.css*/
@import url(fancy.css);
SPAN.NUMBER{
    font-size: 1.5em;
    color: #33f;
    font-style: italic;
}
```

1.1. 出発まで

1.1.1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。
ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「芝刈り」だと誤解している人もいるかもしれません。
「柴刈り」は，暖を取るために，お風呂をたくために，あるいは他の目的のために枯れ枝を拾ってくることです。「柴」は特定の樹種ではなく，低木・雑木林全般を指して使われるのが普通です。
…

1.1.2. 桃太郎の誕生

桃から生まれたので，桃太郎。なんて簡単なネーミングでしょう。しかし，この簡明さこそが名前の本質を表わしています。

図6-10はイメージ図ですが，Netscape Navigator 4.0でもInternet Explorer4.0でも，ほぼこのとおりに表示されます

(3) ブロック化して装飾

SPANは本来インライン表示される要素です．インライン表示である限り，「前後の語句に連続するように表示される」というレイアウトに関する制限から逃れることはできません．

各要素の表示形態はdisplayプロパティによって変更できるのですから，ここでは表示をブロック化し，もうすこし複雑なレイアウトを目指してみましょう．

displayプロパティは，むやみやたらに変更するべきではありません．もし変更する場合は，結果を十分考慮した上で指定してください．

設計が先，記述が後

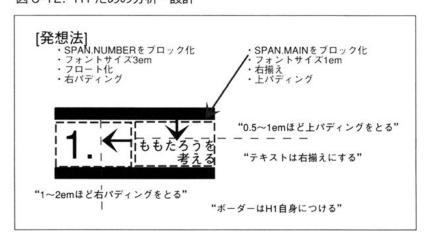
スタイルシートを記述するときには，**あらかじめ目標イメージを明示し**，それを眺めながら必要なプロパティを選び出し，定規などで測りながら値を決定してください．目標を持たずになんとなくプロパティを書き，なんとなく値を記述したところで，好ましい表示結果は得られません．

「分析→設計→実装」とはプログラムを書く際の手順ですが，スタイルシートでも同様です．まず，目標の決定．次に，値の決定．最後に，シートの記述．ただし，「値の決定」には試行錯誤が必要です．

◆H1のための指定例

通し番号のサイズを大き目にし左端に配置し，語句は右側に配置してみましょう（**図6-11, 6-12**）．

図6-11．H1の指定の目標

図6-12．H1ための分析・設計

```
/* book-h1.css */
@import url(atomic.css);
H1{
    font-size: 1.5em;  /*SPAN.NUMBERの大きさを考えて調整*/
    margin: 1em 0em 1.5em 0em;  /*上，右，下，左*/
    padding: 1em;
    background: #ffe;
    border-width: 0.5em 0em;  /*上下，左右*/
    border-style: solid none;
    border-color: #aaf;
    line-height: 1;
}
H1 SPAN{
    display: block;
}
H1 SPAN.NUMBER{
    font-size:3em;
    font-weight:900;
    padding-right: 1em;
    width: 1em;     /*H1の番号は最大でも2文字だと想定し，数字もピリオドも横方向に小さいため，
                     半分の1em程度で十分だろうと考えた*/
    float: left;
}
H1 SPAN.MAIN{
    text-align: right;
    padding-top: 1em;
}
```

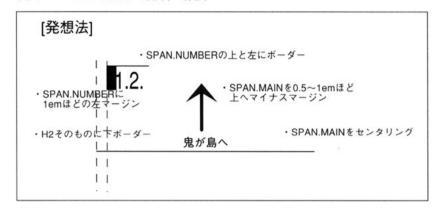
◆ H2のための指定例

語句より少し左上に数値を配置してみましょう（**図6-13，6-14**）.

図6-13．H2の指定の目標

[目標イメージ]

1.2.　鬼が島へ

図6-14. H2のための分析・設計

```
/* book-h2.css */
@import url(atomic.css);
H2{
    font-size: 1.5em;
    margin: 3em 1em 2em 1em;
    border-width: 0em 0em thin 0em;  /*上, 右, 下, 左*/
    border-style: none none solid none;
}
H2 SPAN{
    display: block;
}
H2 SPAN.NUMBER{
    font-size: 1.5em;
    color: #00f;
    margin: 0em auto 0em 1em; /*右マージンを「残り」に指定*/
    width: 3em;     /*H2の番号は最大でも5文字だと想定し, 数字もピリオドも横方向に小さいため,
            半分程度, すなわち3em程度で十分だろうと考えた*/
    border-width: thin 0em 0em 0.5em;  /*上, 右, 下, 左*/
    border-style: solid none none solid;
    border-color: #aaf;
}
H2 SPAN.MAIN{
    text-align: center;
    margin: -0.5em 0em 0.5em 0em;
}
```

それでは, この2つをあわせたスタイルシートを作成しましょう.

```
/* book.css */
@import url(book-h1.css);
@import url(book-h2.css);
H3{
    font-size: 1.3em;
    margin: 2em 0em;  /*上下マージン2em*/
}
H4{
    font-size:1.2em;
    margin: 1em 0em 0em 2em;  /*上マージン1em，下マージン0，左マージン2em*/
}
P{
    text-indent:1em;
    line-height: 1.5em;
    margin: 0.5em 0em;  /*上下，左右*/
}
```

このbook.cssを使った文書の，Netscape Navigator4.0による表示結果を**図6-15**に示します．結果を見て分かるとおり，emユニットの処理がおかしいため「期待通り」とは行きませんが，ほぼ願いはかなえられているといってよいでしょう（巻末資料に，Netscape Navigator4.0とInternet Explorer4.0のCSS1サポート状況を添付します）．

emのかわりにptで記述しなおしたシートを用意すれば，この不具合は避けられます．あまり望ましい行為ではありませんが，完全にサポートされるまでの過渡的な対策として仕方の無いことでしょう．

図6-15．Netscape Navigator4.0でのbook.css

一方，book.cssによる文書のInternet Explorer4.0による表示結果を**図6-16**に示します．テキストに対するフロート指定・width指定などがサポートされていないため望みどおりの結果とはいえませんが，それでも文書構成を伝えることに関してはなんら遜色の無い結果だといえるでしょう．

図6-16. Internet Explorer4.0でのbook.css

1. ももたろうを考える

1.1. 出発まで

1.1.1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

6.4.5. 非対応WWWブラウザへの配慮

これまではスタイルシートを用いて文書スタイルを調整することばかり考えてきました．ところで，**スタイルシートをオフにして**文書を表示してみると，どんな感じになるでしょうか．本書が示した記述方法（HTML文書とスタイルシートの両方）を守っている限り，シート無しでも文書内容の伝達には何の支障もありません．しかし，「見栄え」という意味では，少し**さびしい**表示になるのは仕方ありません．

ところで，この節の例では，見出しをアピールするためにボーダーラインを付加しました．立ち返ってみれば，そのような「区切り線」は，HTMLの**HR要素でも表現できます**．スタイルシートを適切に用いればHR要素は必要ないのですが，それでもしかし，非対応WWWブラウザのためにHR要素を付け足してみてはいかがでしょうか．

適切にHR要素がマークアップされていれば，スタイルシートをオフにしてもある程度の見やすさ・読みやすさを確保できます．筆者は，仕切り線を想定した部分にはHR要素を記述することをお勧めします（**図6-17**）．そして，スタイルシート対応WWWブラウザに余分な表示をさせないために，スタイルシート中に

```
HR { display: none}
```

と加えておきましょう．

以降では，atomic.css にこの宣言が書かれているものとします．

図 6-13．スタイルシートオフ時のみ HR 要素を表示

1.1 出発まで

1.1.1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

ところで，最近の人は「柴」をご存知でしょうか。柴とは無関係の生活をしているため，「芝刈り」だと誤解している人もいるかもしれません。

「柴刈り」は，暖を取るために，お風呂をたくために，あるいは他の目的のために枯れ枝を拾ってくることです。「柴」は特定の樹種ではなく，低木・雑木林全般を指して使われるのが普通です。

...

1.1.2. 桃太郎の誕生

桃から生まれたので，桃太郎。なんて簡単なネーミングでしょう。しかし，この簡明さこそが名前の本質を表わしています。

Transitional の属性を利用する

スタイルシート非対応 WWW ブラウザを使っている人のために，（HTML4.0-Strict では破棄されたものの，Transitional では残されている）いくつかのスタイル規定のための属性を用いるのは悪いことではありません．たとえば，H1 ～ H6 要素や P 要素の ALIGN 属性を text-align プロパティの代替にすることは悪いことではありません．

ただし，Transitional の属性を使う際には，次の 2 点を守ってください．

❶DOCTYPE 宣言で 4.0-Transitional であることを明示する
❷スタイルシートを利用できない人に，スタイルシートの内容を伝える目的で，HTML の用意した属性を利用する

すなわち，「BODY 要素で色指定をしたから，スタイルシートでは色を指定しなくても良い」「HTML が用意している属性で指定できない部分だけスタイルシートを使えば良い」などとは考えないでください．そのように考えてしまっては，**本末転倒**です．Transitional の意味は，「スタイルシートが普及するまでの過渡的手段」なのですから．

6-5 「一般性」と「特殊性」のバランス

6.5.1. "DIV.MAIN" = "P + UL + BLOCKQUOTE"

ここでは，特定の文脈を対象にしたスタイル指定を調整する際に必要なテクニックをお伝えします．例として，本文が段落（P要素）・リスト（UL要素，OL要素）・引用ブロック（BLOCKQUOTE要素）によって構成されているケースを用います．

この3者の左マージンを，PとULは見出しに対して3文字分，BLOCKQUOTEはPやULよりさらに2文字分設定するにはどうすればいいでしょうか（図6-18）．

図6-18. 複数の要素からなる本文の左マージン目標図

```
<BODY>
<H1>私のPC</H1>
<P>私のPCの宣伝文句は，かの長島茂男さんがにこやかに語っていました．</P>
<BLOCKQUOTE>
    <P>「サポート体制で選べば，フローラ.」</P>
</BLOCKQUOTE>
<P>そう，日立のフローラです．</P>
<P>1994年に購入したため，いまやローエンドなスペックです．</P>
<UL>
    <LI>Pentium75MHz
    <LI>RAM48MB（標準は16MB）
    <LI>ハードディスク512MB
</UL>
</BODY>
```

単純に考えれば，各要素に必要なマージンを直接設定すれば実現できます．

```
P, UL{ margin-left: 3em}
BLOCKQUOTE{ margin-left: 5em}
```

しかし，これでは「BLOCKQUOTEはPやULよりも2文字多い」という関係が理解しにくくなってしまいます．さらに，P・ULに対するマージンを変更したときには，関係を維持するためにBLOCKQUOTEのマージンも変更する必要があり，**大変メンテナンス性が悪い**といえます．

これは，文書構成が十分に示されていないために起こる混乱です．マージンは「親要素のマージンにどれだけ追加するか」という形で指定するため，構成が不明確だと指定に不整合が生じやすくなります．「段落・引用ブロック・リストが合わさって狭義の本文になっている」という構成をより明確にするために，DIVブロックを追加しましょう（**図6-19**）．

図6-19．本文の構造ツリー

```
<BODY>
<H1>私のPC</H1>
<DIV CLASS="MAIN">
    <P>私のPCの宣伝文句は，かの長島茂男さん…</P>
    <BLOCKQUOTE>
        <P>「サポート体制で選べば，…」</P>
    </BLOCKQUOTE>
    <P>そう，日立のフローラです．</P>
    <P>1994年に購入したため，いまや…</P>
    <UL>
        <LI>Pentium75MHz
        <LI>RAM48MB（標準は16MB）
        <LI>ハードディスク512MB
    </UL>
</DIV>
</BODY>
```


構成に沿って考えると，「見出しに対し3文字分の左マージン」を指定するべき対象はDIV.MAINなのだと思い当たります．PやULはそのマージンを修正せずに用いていますが，BLOCKQUOTEはさらに2文字多いマージンが欲しかったのです．

```
DIV.MAIN { margin-left: 3em}
DIV.MAIN BLOCKQUOTE{ margin-left: 2em}  /*DIV.MAINよりも2em多い*/
/*仮にDIV.MAINの左マージンを変更したとしても，BLOCKQUOTEとの位置関係は不動*/
```

このように，**構成を明確にすればするほど**スタイルシートの内容も明確になり，メンテナンス性も向上します．

6.5.2. 「特定の文書への対応」と「一般性確保」のジレンマ

6.4.1.では「狭義の本文」を「PとBLOCKQUOTEとUL」としましたが，場合によってはPREが入るかもしれませんし，別のDIVが入り込むかもしれません．逆に，BLOCKQUOTEの無い文書もあるでしょう．すなわち，HTML文書の構成は文書型定義で一般的に定義されていますが，その定義は大変に大まかなもので，実際の文書インスタンスの構成は全く特定できないのです．

すると，**図6-18**のスタイルシートは，この文書にはぴったり適用できますが，ほかの文書もうまく機能するとは限りません．そもそも，わざわざDIV.MAINをマークアップしたHTML文書でなければ**図6-18**のスタイルシートは活用できないのです．

「ある特定の文書に最適のシート」を追い求めることと「一般性のあるシート」を追い求めることは，基本的に両立できません．

6.5.3.ジレンマと上手に付き合う方法

両立できないものを無理に両立させようと努力するよりも，これらを分けて構築するほうが得策です．以降では，「どのように分けるべきか」について考えていきます．

(1)「一般」と「特殊」を分離する

ところで，「一般性のあるシート」－多くのHTML文書に適用できるシート－とはどのようなものでしょう．端的には，基本的な要素だけへの宣言で構築した（すなわち，クラス，DIV，SPANに頼らない）シートだと考えられます．本書で紹介したstandard.cssやfancy.cssはこれに当たります．

クラス・DIV・SPANを活用したシートを作る場合は，fancy2.cssのように「一般性のあるシートをimportして，特殊指定を追加する」ように書くのが賢いと思われます．なぜなら，想定した要素が無い場合は単にimportしたもとのシートとして機能するからです．

すなわち，うまくジレンマと付き合うには，「基本的な要素だけへの宣言」と「特殊なクラスや要素への宣言」を別シート（別ファイル）として構築しておくべきだと考えられます．そのようにして作られたシートを必要に応じて組み合わせていけば，常に一般的な効果を期待できます．

（2）ごく特殊なものはSTYLE要素に押し込む

ところで，シートが使いまわせるかどうかは，「想定するクラス・DIV・SPANを利用した特殊な要素が，文脈上，親要素になるかどうか」である程度判断できます．

たとえば，fancy2.cssやbook.cssのように「期待する特殊な要素が，文脈上，単なる子要素である」場合には，期待する要素が存在しない場合でも，ある程度整合性の取れた表示になります（**図6-20**）．なぜなら，「その要素が無いこと」の影響が一部にとどまるからです．したがって，このシートは「ある程度使いまわしが効く」と判断できます．

図6-20．book.cssをSPAN指定の無い文書に適用した場合

1. ももたろうを考える

1.1. 出発まで

1.1.1. はじまりはじまり

桃太郎のお話は，おとぎ話に良くあるように，「むかしむかし，あるところに」で始まります。おじいさんは山へ柴刈りに，おばあさんは川へ洗濯に行きます。

ところが，期待する特殊な要素が親要素となる場合（たとえば**図6-19**）は上手くいきません．なぜなら，「その要素が無いこと」の影響が別の要素にも波及するからです．

もし，**図6-19**のように「狭義の本文は，つねにDIV.MAINと指定する」というルールを（自分独自のルールとして）常に守るのであれば，このようなシートも使いまわしが効きます．しかし，そうでないかぎり，そのシートは1回限りしか利用されないでしょう．ということは，特定のHTML文書を対象にしたスタイルシートは，一つのファイルとして保存するのではなく，はじめからHTML文書にSTYLE要素として書き込んでおく（別のHTML文書から利用できないようにする）のが賢いとおもわれます（逆に，使いまわしの効くシートは必ず独立したファイルとして保存しましょう）．

どちらにせよ，**もっとも重要**なのは「一般指定と特殊指定をひとつのシートに混在させないこと」だと言えます．“二兎を追うものは一兎も得ず”です．

6-6 「一般性のあるシート」の例

standard.cssやfancy.cssはそのままでも十分実用性のあるスタイルシートですが，実は本書の印刷代金との兼ね合いで^^;，白黒でも内容がはっきり伝わるように考慮した結果，色や画像をあまり使っていません．WWWでは色や画像を気軽に使えるため，逆に，このままでは物足りなくなると思われます．

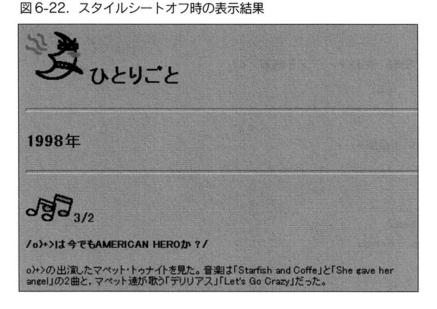
筆者は，atomic.cssに幾つかのボーダー・色指定・背景画を追加したものを自分用の標準スタイルシートとして利用しています．参考までに，それをここで公開します．

6.6.1. 「ks.css」

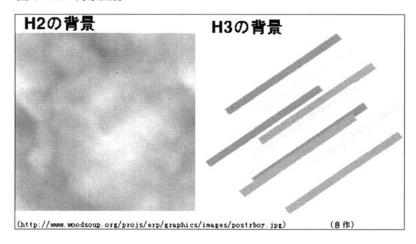
（1）表示結果

図6-21．ks.cssの表示結果

```
<H1><IMG SRC="moon.gif" ALT="">ひとりごと</H1>
<HR>
<H2>1998年</H2>
<HR>

<H3><IMG SRC="music.gif" ALT="">3/2</H3>
<H4>/o)+&gt;は今でもAMERICAN HEROか？/</H4>
<P>o)+&gt;の出演したマペット・トゥナイトを見た．音楽は「Starfish and Coffe」と
「She gave her angel」の2曲と、マペット達が歌う「デリリアス」「Let's Go Crazy」だった．</P>
```

図6-22. スタイルシートオフ時の表示結果

（2）用意した背景画像

スタイルシートはあくまで表示の調整の道具であり，画像の代替になるものではありません．過度な画像埋め込みは望ましくないものの，スタイルシートだけで全ての表現効果を得ようとするのも賢いことではありません．適切に画像を利用しましょう．

図6-23. 背景画像

（3）スタイルシート

```
@import url(standard2.css);
@import url(ie-back.css);
@import url(ks-back.css);
BODY{
    color: black;
```

```
    background: #ffe;
}
H2{
    margin: 2.5em 2em;  /*上下，左右．左右マージンを追加．*/
    border-width: 0em 0.5em;
    border-style: none solid;
    border-color: #aaf;
    padding: 1em 2em; /*パディングを追加*/
}
H3{
    width: 100%;
    border-width: thin 0em 0em 0em;
    border-style: solid none none none;
    padding: 1em 0em;
}
BLOCKQUOTE{
    margin: 2em;
    border-width: 0.3em 0em ; /*上下，左右*/
    border-style: dotted none;
    border-color: #f88;
    font-family:monospace;
    padding: 0.8em 1em;
}
```

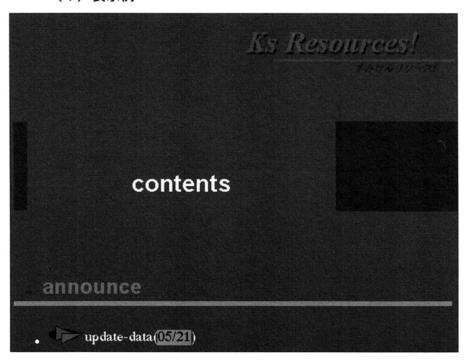
```
/*ie-back.css*/
P STRONG{
    color:black;
    background: #faa;
    font-weight: bolder;
}
A:link STRONG{background:#ccf;}
A:visited STRONG{background:#fcc;}
A:active STRONG{background:#f88;}
LI{margin-bottom:0.5em;}
```

Netscape Navigator4.0はbackground系の処理が中途半端なため，まったくサポートしていないものよりもマズイ結果を生み出すことがあります．そこで，Netscape Navigator4.0が@importを解釈できないという不具合を逆に利用し，別シートとして記述しました．これならば，すでに正常に機能するInternet Explorer4.0には支障ありませんし，後にNetscape NavigatorがCSSを完全サポートしたときにも上手く機能します．とはいえ，このテクニックは，あくまでも過渡的なものです．

同様に，Netscape Navigator4.0 は，LI 要素への指定を「マーカー」への指定であると勘違いするため，おかしな表示結果になってしまいます．

```
/*ks-back.css*/
H2{background: url(../back2.jpg);}
H3{background: url(../back3.gif) right center repeat-y;}
```

6.6.2. 「prince.css」

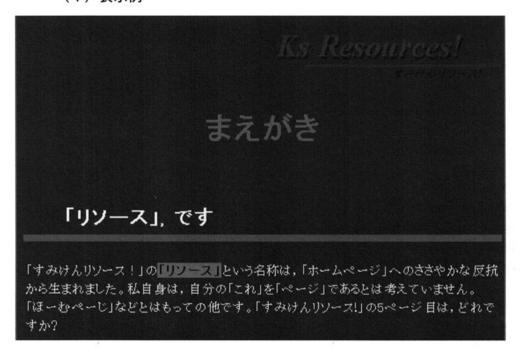
（1）表示例

（2）スタイルシート

```
/*prince.css*/
@import url(atomic.css);
@import url(ie-back.css);
@import url(prince-back.css);
BODY{
    background:#646;
    color:#fed;
}
:link{color:#0af}
H1{
    font-size:3em;
}
```

```
H2{
    font-size:2em;
    border-width:0em 5em 0em 0.5em;
    border-style: none solid;
    border-color: #a08;
    padding: 2em 0.5em 1em 0.5em;
    margin: 2em 0em 0.5em 0em;
}
H3{
    font-size:1.8em;
    margin: 3em 0.2em 1em 0em;
    color: #99c;
    border-bottom-style: solid;
    border-bottom-width:medium;
    border-color: red;
    padding: 0em 0em 0.3em 0.5em;
}
H4{
    font-size:1.3em;
    margin: 1.5em 0em 0.5em 0.5em;
}
H5{
    font-size:1.1em;
    margin: 0.5em 0em 0.5em 0.5em;
}
.CAUTION{
    margin: 1em 1em 1em 20%;
    border-width: 0em 0.5em 0em 3.5em;
    border-style: none solid;
    border-color:#f22;
    padding: 2em 0.5em 0.5em 1em;;
}
BLOCKQUOTE{
    margin: 2em;
    font: italic 1em monospace;
    color: #bbb;
    border-width: 0em thick;
    border-style: solid;
    border-color: #a40;
    padding: 1em 0.5em;;
}
```

```
/*prince-back.css*/
P STRONG{
    color: #404;
    background: #a4a;
}
.SAD{
    color:red;
    background:transparent;
    font: 900 italic 1.2em monospace;
}
```

6.6.3. 「zappa.css」

（1）表示例

（2）スタイルシート

```
/*zappa.css*/
@import url(ie-back.css);
BODY{
    color:#fed;
    background: #555;
}
:link{color:#0af}
```

```
H1{
    font-size:3em;
}
H2{
    font-size:2.4em;
    margin: 1em 2em;
    color: #d9d;
    text-align :center;
}
H3{
    font-size:1.6em;
    border-width: 0em 0em thick 0em;
    border-style: solid;
    border-color: #888;
    padding: 0em 0em 0.5em 0.5em;;
    margin: 2.5em 0em 1em 0em;
}
H4{
    font-size: 1.3em;
}
H5{
    font-size: 1.1em;
}
BLOCKQUOTE{
    margin:1.5em 1em 1.5em 3em;
    border-width: 0em 0em 0em thick;
    border-style: none none none solid;
    border-color: #969;
    padding: 1em 1em 1em 2em;
}
H4{margin: 2.5em 0em 0em 1em}
H5{margin: 2em 0em 0em 1.3em}
H6{margin: 2em 0em 0em 1.5em}
```

6-7 終わりに

6章では，スタイルシートを記述するための発想法を紹介しました．本書の指針に従えば，HTML文書にもスタイルシートにも無理の無い形で両者を連携させられるようになります．

スタイルシートは，HTMLと比べて「はじめの一歩」を踏み出すまでに必要となる知識が多いためか，まだあまり利用されていないのが現状です．しかし，スタイルシートは実に魅力的な道具です．ぜひスタイルシートについて学び，活用してみてください．本書の内容が皆様の「はじめの一歩」の手助けになることを願います．

▶ 考え方のまとめ

・先に文書の構造・構成を明確にしましょう．同時に，HTML文書からスタイル情報を排除しましょう．

・スタイルは，構成をわかりやすく（本文は本文らしく，見出しは見出しらしく）することを目的に指定しましょう．

・スタイルは，文書構造にしたがって，「一般 → 例外」の順に指定しましょう．

・対応していないWWWブラウザ利用者のことも考え，スタイルシートをオフにしたときにもわかりやすいかどうかをチェックしましょう．

・一般性のある指定と特定の文書用の指定を分離しましょう．前者は独立したファイルに保存し，後者は対象にしたHTML文書内に埋め込みましょう．

▶ 技術面のまとめ（指定決定までの手順）

・目標の表示結果を明確にする．

・理想図から，文書構成を読み取る．（この際に，要素ごとの表示範囲を図に書き込むとよりわかりやすくなる．）もし文書マークアップが不十分であると判明したら，追加・修正する．

・各要素のフォントサイズ，マージン，パディングなどを計測する．できる限りemユニットやパーセントによる値を考える．

・設計にしたがって，スタイルシートを記述する．

・場合に応じて，シートの分割や取り込みを試みる．

new feature in CSS2

7章では，CSS2で追加された新たなプロパティを紹介します．CSS2で追加された印刷時の考慮，音声再生の考慮によって，HTML文書はWWWのみならず広範囲に有効な文書フォーマットになるでしょう．

総評ですが，CSS2で追加された新たな側面は，全般に扱いの難しい，どちらかといえばプロフェッショナル向けのものばかりです．もっとも，CSS1が一般的なプロパティを網羅しており，CSS2はそれの上位互換として足りなかった部分を追加したのですから，その狙いが高度になっているのは当然だといえます．

7-1 表示の位置決め

7.1.1. 導入された概念

▶viewport（覗き窓）

スクロール可能なメディアで，実際に表示されている部分を指す概念です．

▶containing block（コンテナブロック）

ある要素に注目した場合に，その要素のブロック系親要素の中でもっとも近い親を指す概念です．マージンやパディングの計算で利用されます．

▶Anonymous block boxes（匿名ブロック）

ブロックコンテナ系要素に直に記述されたインライン系要素が存在する時に，親子バランスを取るために挿入されるコンテナブロックです．

HTML3.2および4.0Transisionalでは，ブロックを含みうる要素（DIV・BLOCKQUOTE・BODYなど）の子要素は「(%inline; | %block;)*」であり，**図7-1**のような記述も許されます．

図7-1．anonymous block boxのある記述

```
<BODY>
    事前に<A HREF="maegaki.html">前書き</A>を読んでください．
    <H1>第1章：プリンスとの出会い</H1>
    それは，岡村靖幸との出会いから始まる．
    <P>岡村靖幸は…
</BODY>
```

しかし，# PCDATAの位置的な親子バランスを考えた場合，**図7-2**の記述のほうが望ましいといえます．

図 7-2. anonymous block boxの無い記述

```
<BODY>
    <P>事前に<A HREF="maegaki.html">前書き</A>を読んでください.
    <H1>第1章：プリンスとの出会い</H1>
    <P>それは，岡村靖幸との出会いから始まる.
    <P>岡村靖幸は…
</BODY>
```

すなわち，**図7-1**にはBODYの直接の，子要素としてインライン系要素（# PCDATA）が存在しています．表示整形上，この#PCDATAのコンテナブロックをBODYと考えるよりも，見えないブロック系要素がコンテナになっていると考えたほうが得策です．その，存在しないコンテナブロックを「匿名ブロック」と呼びます．

▶Anonymous inline boxes（匿名インライン）

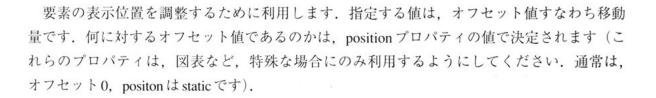
普通のブロック系要素の子要素のうち，インライン系要素にラップされていない直の#PCDATAを指す概念です．

「<P>Some <EM>emphasized</EM> text」のうち，「Some 」と「 text」のこと．

7.1.2. 位置決めの概念

※ top，right，bottom，left　オフセットプロパティ

値	<一般サイズ>｜<パーセント>｜auto
初期値	auto
適用対象	位置指定される要素（注：displayプロパティがstatic以外?）
パーセントの意味	コンテントブロックのwidth/heightに対する割合
継承される？	no

要素の表示位置を調整するために利用します．指定する値は，オフセット値すなわち移動量です．何に対するオフセット値であるのかは，positionプロパティの値で決定されます（これらのプロパティは，図表など，特殊な場合にのみ利用するようにしてください．通常は，オフセット0，positonはstaticです）．

※ position　位置決めアルゴリズム

値	static | relative | absolute | fixed
初期値	static
適用対象	自動生成以外の全要素
継承される？	no

オフセット指定の適用方法を指定します．各キーワードの意味は**表 7-1** のとおりです．

表 7-1．positon プロパティのキーワードの解説

キーワード	解説
static	通常の指定．left プロパティと top プロパティは適用されない（注： right と bottom は?）．
relative	static で表示されるべき位置からのオフセットで表示位置を決定する．
absolute	コンテントブロックからのオフセットで表示位置を決定する．
fixed	連続メディア（スクロールメディア）では，viewport に対する位置として絶対位置が計算され，スクロールに流されずに常に表示される．ページメディアでは，ページに対する位置として絶対位置が計算される．

仕様書では各指定に関する細かい事例を取り上げて解説していますが，本書ではそこまでは踏み込みません．

※ z-index　重ねあわせ表示の優先順位

値	auto | < 整数 >
初期値	auto
適用対象	位置指定される要素（注： display プロパティが static 以外?）
継承される？	no

表示が重なった場合の「背後に移動」「前面に移動」の順序付けを指定します．数値が小さいほど背後に表示されます．

7.1.3. 記述の流れ

※ direction　文章の流れる方向

値	ltr | rtl
初期値	ltr
適用対象	全要素
継承される？	yes

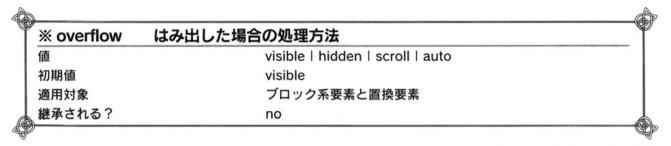
国際化対処の一環で，記述の流れが「左→右（left to right: ltr)」であるか「右→左（right to left: rtf)」であるかを指定します．この値によって，左右マージンなどのauto時の計算方法などが調整されます．

※ unicode-bidi　　　ユニコード

ユニコード処理に関係するプロパティですが，本書では省略します．

7.1.4. 表示内容の調整

※ overflow　　はみ出した場合の処理方法

値	visible I hidden I scroll I auto
初期値	visible
適用対象	ブロック系要素と置換要素
継承される？	no

次のような場合には，表示すべき内容が表示枠（画面，ページ，あるいは指定した表示枠）からはみ出してしまう可能性があります．

- 絶対位置指定で表示している場合
- マイナスのマージンを指定した場合
- width や height を明示した場合
- white-space プロパティで pre 指定した場合

overflow プロパティは，はみ出してしまった部分の処理方法を指定するプロパティです（表 7-2）.

表 7-2. overflow プロパティのキーワードの解説

キーワード	解説
visible	はみ出してでも表示（枠を破棄しても構わない）
hidden	見えない部分はそれで構わない（見えないままにしておく）
scroll	スクロールバーなどを用いることで見えるようにする （小さな viewport を生成する）（印刷などの場合は，visible 処理する）

visible 以外の指定では，clip プロパティで表示すべき領域の形状を指定します．しかし，CSS2 の段階ではまだ十分な検討が済んでいないようですので，ここでは省略します．

※ visibility 表示するか否か

値	visible | hidden | collapse /* 表示する，しない，しない（TABLEのrowやcolumnのみ）*/
初期値	inherit
適用対象	全要素
継承される？	no

要素を表示するか否かを指定します．「display: none」では要素を存在しないものと処理しますが，「visibility: hidden」では「要素は存在するが表示しない」ことになります．具体的には，「visibility: hidden」では，その要素を表示するために必要なだけの空白が表示されます．

collapseはTABLE用のhiddenであるようですが，仕様書では具体的な差異が分かりませんでした．

この指定を「:hover」疑似クラスなどと組み合わせると，ちょっとしたトリッキーな効果を得られます．ただし，やりすぎにはご注意を!

```
DIV.HIDE A{visibility: hidden}
DIV.HIDE A:hover{visibility: visible}

<DIV CLASS="HIDE">
<P>あなたを<A HREF="http://www.asahi-net.or.jp/~jy3k-sm/i_net/books.html"
アヤシイ世界</A>に案内します.
</DIV>
```

このアンカーの文字はマウスをかざしたときしか表示されない．

7-2 自動生成文字

7.2.1. content プロパティと「:before」「:after」疑似要素

※ content	内容
値	[< 文字列 > | <URL> | < カウンタ > | attr(X) | open-quote | close-quote | no-open-quote | no-close-quote]+
初期値	"" /* 空文字 */
適用対象	:before 疑似要素と:after 疑似要素

「:before」疑似要素と「:after」疑似要素の具体的な内容を指定するプロパティです．任意の文字列，画像やサウンド（を指すURL），カウンタ，引用符，およびその組み合わせが指定可能です．なお，引用符とカウンタについては後述します．

```
@media aural{
    H1:before{
        content: url(chime.au);
    }
}
H1:before{
    display: inline;
    content: url(good.gif);
}
STRONG:before, STRONG:after{content : "*"}
BLOCKQUOTE:before{content:"(引用開始) "}
BLOCKQUOTE:after{content:"(引用終了) "}
Q:before{content: open-quote}
Q:after{content: close-quote}
```

attr(x)は，HTML文書における属性値をcontentとして表示させるためのキーワードです．その属性に値が無い場合は空文字が表示されます．「A:before {content: "[" attr(REL) "]"}」であれば，**図7-3**のように表示されます．

＜文字列＞には""が必要であることに注意してください．

図 7-3. attr(X)利用例

```
<STYLE TYPE="text/css">
A:before {content: "[" attr(REL) "]"}
</STYLE>
<P>目次
<UL>
    <LI><A HREF="1.html" REL="NEXT">第１章：プリンスとの出会い</A>
    <LI><A HREF="2.html">第２章：情報を追い求めて</A>
    ......
    <LI><A HREF="appendex.html" REL="APPENDIX">索引</A>
</UL>
```

目次
・[NEXT]第 1 章：プリンスとの出会い
・[]第 2 章：情報を追い求めて
……
・[APPENDIX]索引

改行

CSS1 では BR 要素を表現するプロパティは用意されませんでした．このことは課題として残されていたのですが，ついに CSS2 で実現できるようになりました．

```
BR:before { content: "¥A"}
```

文字列の ¥ はエスケープシーケンス用のメタキャラで，¥A は改行文字なのです!
…な，なんだかインチキ臭いんですが，ともかくこれで実現可能です．

7.2.2. 引用符

▶ content プロパティにおける引用符指定

・open-quote，close-quote

quotes プロパティで指定した「引用符開く」「引用符閉じる」の記号が表示されます．引用ネストの深さに応じた引用符が自動的に選択されます．

・no-open-quote and no-close-quote

引用符を表示せずに，引用ネストの深さだけを変更します．

```
Q:before { content:open-quote }
Q:before { content:close-quote }

<P>ボブディランは<Q>30才以上のヤツを信用するな</Q>と言った.
```

ボブディランは「30才以上のヤツを信用するな」と言った.

※ quotes　引用符

値	[< 文字列 > < 文字列 >]+ | none /* 開く　閉じる　開く　閉じる… */
初期値	実装依存
適用対象	全要素
継承される？	yes

content プロパティの引用符指定に呼応する引用符を指定します.「<ひらく><閉じる>」のセットでのみ指定できます. 複数の引用符を指定すると, 引用ネストの深さに応じた引用符を設定したことになります.

```
quotes: "{"  "}"    "["  "]"     "("  ")";
{[()]}
```

カンマ区切りではなく, 空白区切りです.

なお, 5章で紹介した:lang疑似クラスを用いれば, 言語の違いを考慮した指定が可能です.

```
Q:lang(en) { quotes: '"'  '"'    "'"  "'" } /*English*/
Q:lang(ja) { quotes: "「"  "」"    '“' '”' }   /*Japanese*/
```

文字列として「"」を記述したい場合は, 全体を"で囲むことになります.

余談ですが, 必ずしも伝統的な引用符を用いる必要はありません.

```
BLOCKQUOTE{quotes:"（引用開始）"  "（引用終了）"}
```

content プロパティの指定抜きで quote プロパティを指定しても, 引用符は表示されません. このことには十分ご注意ください.

7.2.3. カウンタ（通し番号）

▶content プロパティのカウンタ指定

文書記述者は，「カウンタ名」を記述することでその名前のカウンタを利用できるようになります．プログラミングにおける変数宣言のようなものはなく，以降の例のように「カウンタ名」を利用し始めるだけで，自動的に対応するカウンタが作成されます．カウント制御はカウンタ名ごとに独立して行われます．なお，「カウンタ」はいくつでも作成できます．

カウンタ値を表示するには，content プロパティの値として「count(<カウンタ名>)」を指定します．

「カウンタ名」に利用できる文字は，ラテンアルファベットの「a-z」「A-Z」，数字「0-9」それにハイフン「-」のみです．始めの1文字はラテンアルファベットのみです．厳密には，ISO 10646 charactersの161番以上も許されます．

```
H1:before{
  content: counter("chapter-num") "章：";
}
```

ただし，これだけではカウンタの値が決定されません．「カウントアップ」と「値のリセット」には次のプロパティを用います．

※ counter-increment　増減させるカウンタの指定

値	[<カウンタ名> <整数>?]+ | none /* カウンタ名，カウントアップ値 */
初期値	none
適用対象	全要素
継承される？	no
マイナス値	可

このプロパティが指定されていると，その要素が現れるたびに，記述したカウンタが指定した値だけ増減します．カウントアップ値を省略した場合は，「1」が指定されます．

※ counter-reset	リセットするカウンタの指定
値	[< カウンタ名 > < 整数 >?]+	none /* カウンタ名，リセット値 */
初期値	none
適用対象	全要素
継承される？	no
マイナス値	可

このプロパティが指定されていると，その要素が現れるたびに，指定したカウンタの値が指定値にリセットされます．リセット値を省略した場合は，「0」にリセットされます．

▶ **利用例**

「chapter-num」と「section-num」の2つのカウンタを利用してみましょう（CSS2仕様書12.5を改定）．

```
H1:before {
    display: inline;
    counter-increment: "chapter-num";  /* カウンタ「chapter-num」に1加算 */
    content: counter("chapter-num") "章：";
    counter-reset: "section-num";  /* カウンタ「section-num」を0にリセット */
}
H2:before {
    display: inline;
    counter-increment: "section-num";  /* カウンタ「section-num」に1加算 */
    content: counter("chapter-num") "." counter("section-num") "：";
}

<H1>はじめに</H1>
<H2>Frank Zappaとは</H2>
<H2>そのアルバムの数</H2>
<H1>ZAPPAアルバム全紹介</H1>
```

```
1章：はじめに
1.1 ：Frank Zappaとは
1.2 ：そのアルバムの数
2章： ZAPPAアルバム全紹介
```

カウンタ値の表示，カウントアップ，リセットの仕組みが独立しているため，「カウントアップだけして表示しない」「表示だけしてカウントアップしない」という指定も自由自在です．必要に応じてお試しください．

カウンタのネストとスコープ

カウンタを利用するのは，見出しやリストに通し番号を付与する時が大部分だと思われます．例では見出しのみを取り上げましたが，OL要素のカウントアップも上記の説明と同じようにして記述可能です．

ところで，リストのネスト時の処理はどうなるのでしょう．構造化プログラミングなどを経験していない人には分かりにくいところだとは思いますが，「リストにおけるカウンタのスコープは，リストのネスト状況に合わせてネストします」というのが答えです（次例のソースは，CSS2仕様書12.5.1を改定したものです）．

```
OL {counter-reset: item}
LI:before {
    content: counter(item) ". ";
    counter-increment: item
}

<P>好きな食べ物
<OL>
    <LI>麺
    <OL>
        <LI>うどん
        <LI>スパゲッティ
    </OL>
    <LI>肉
    <OL>
        <LI>鶏肉
        <LI>牛豚合挽
    </OL>
</OL>
```

```
好きな食べ物
1.麺
  1.うどん
  2.スパゲッティ
2.肉
  1.鶏肉
  2.牛豚合挽
```

▶「a，b，c…」なカウント

数値ではなく，lower-romanやkatakanaでカウントアップしたい場合は，contentプロパティの値として「counter(＜カウンタ名＞，＜list-style-type＞)」を指定します（カンマ区切りです）．この場合は，「1,2,3……」ではなく，「a,b,c……」「い,ろ,は……」とカウントアップされます．

7-3 ページメディア

CSS2の段階では，ごく基本的な印刷制御プロパティが用意されただけで，数多くのネタが「今後の課題」になっています．以降にあげるネタには未対応です（ページメディアといっても印刷とは限らないのですが，一般には「印刷」だと受け止めても支障はないでしょう）．

・一枚の紙面を拡大して数ページに割り付ける（n-up）
・数枚の紙面を縮小して1ページに割り付ける（tiling）
・段組
・柱やノンブル（すべてのページに存在する「第何章」とか「ページ数」）

7.3.1. @pageセレクタ

ページそのもののサイズなどを指定するために，セレクタ「@page」を導入します．「@page」セレクタにはサイズとマージンを指定できますが，ボーダーやパディングは今後の課題になっています．各プロパティのパーセント指定は，紙のサイズを指定するsizeプロパティからの相対記述になります．なお，emユニットは使えません．また，ページへの指定は継承の計算に組み入れません．

とくに印刷時の左右ページを指定する場合には，「@page:left」「@page:right」疑似クラスを使用します．さらに，「最初のページ」として「@page:first」が用意されています．

```
@page { size 8.5in 11in; margin: 2cm }
@page:left { margin: 2cm 2cm 2cm 3cm }/*上,右,下,左*/
@page:right { margin: 2cm 3cm 2cm 2cm }
```

7.3.2. 「@page」セレクタ用プロパティ

※ size 紙面サイズ

値	< 絶対サイズ >{1,2} l auto l portrait l landscape /*width height ：一方のみ指定時は正方形に */
初期値	auto
適用対象	ページ

紙面サイズを指定します．＜絶対サイズ＞以外は，すべて相対指定です（**表7-3**）．

表7-3. size プロパティのキーワードの解説

キーワード	解説
portrait	現在の紙サイズで，縦置き
landscape	現在の紙サイズで，横置き
auto	現在の設定のまま

（注：「現在の紙サイズ」とは，HTML文書を読み込んだコンピュータシステムの現在の設定を指しています）

はみ出したら?

紙サイズの指定によっては，文書内容がページに収まらない可能性があります．仕様書には，こういう場合にWWWブラウザが行うべき処理までが書かれています．

1：紙の縦横をかえてみる
2：サイズをかえる

それでも駄目であれば，はみ出した部分は印刷されないことになります．

余談ですが，紙サイズよりもページ設定サイズが極端に小さい場合は，WWWブラウザは自動的に「真ん中寄せ」などの処理をするだろう，とも書かれています．

※ marks　トンボ

値	[crop ll cross] l none /* 裁ち切りトンボ　位置あわせトンボ */
初期値	none
適用対象	ページ

組版で言うところの「トンボ（ページものの四隅にある1ページの範囲を示す位置決めマーク)」です（**表7-4，図7-4**）．なお，その形状，サイズなどはWWWブラウザに一任されています．

表7-4. markプロパティのキーワードの解説

キーワード	解説
crop	裁ち切り位置を指定するトンボ
cross	複数の紙を重ねる時の位置決めスポットを指定するトンボ

図7-4. トンボ

7.3.3. ページメディア専用プロパティ

※ page-break-before, page-break-after　改ページ

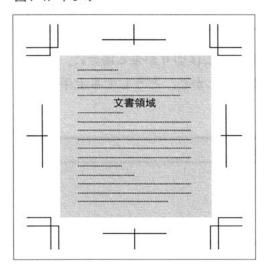
値	auto | always | avoid | left | right
初期値	auto
適用対象	ブロック系要素
継承される?	no

その要素の「始め」「終わり」で改ページするかしないかを指定します（表7-5）.

表7-5. page-break系プロパティのキーワードの解説

キーワード	解説
auto	WWWブラウザに一任
always	常に改ページする
avoid	改ページを避ける
left	1あるいは2ページ分改ページして，左ページへ
right	1あるいは2ページ分改ページして，右ページへ

出版における慣例では，横書きでは「大見出しによる大扉」を必ず右ページ（縦書きでは必ず左ページ）に配置し，中見出しでは優先的に改ページします．

※ page-break-inside　泣き別れ	
値	auto	avoid
初期値	auto
適用対象	ブロック系要素
継承される?	yes

「page-break-inside: avoid」では，指定した要素の最中にページ区切りが来る場合に，無理にそのページに押し込むか，全体を次ページに送り込みます．しかし，どうしても上手く処理できないケースも存在しますのであしからず．

※ orphans，widows　泣き別れ条件（孤児，やもめ）	
値	＜整数＞
初期値	2
適用対象	ブロック系要素
継承される?	yes

「orphans（孤児）」とは，組版用語で「前ページに残す行」のことです．また，「widows（やもめ）」とは，「次ページに送る行」です．各プロパティには，それぞれの泣き別れの「最小値」を指定します．

WWWブラウザは，page-break系プロパティの指示にしたがって改ページ処理をした後，この指定を参考にページ送りを調整します．すなわち，orphansやwidowsが指定値以下のブロックが存在する場合には，要素全体を次のページに送り込むなどの処理をします．

泣き別れは，position指定がabsoluteおよびfixedの場合は考慮されません．absoluteやfixedでは「ページからはみ出した」場合の処理に準じ，端的には「はみ出した部分は印刷しない」ことになります．

仕様書には，さらに細かい処理の指針が提示されているのですが，それらはWWWブラウザへの指示であるために，ここでは省略します．

7.3.4. 名前付き「@page」と「page」プロパティ

▶ 名前付き「@page」

図表などの表現のために，印刷途中で**ページの縦横を変更**したり，**サイズを変更**したいことがあります．そのために，名前付き「@page」を用意し，文章中でページを切り替えられるようになっています．

名前付きページは「@page <ページ名>」（空白区切り）で記述します．各要素をどの@pageで印刷するのかは，pageプロパティで指定します．

「ページ名」に利用できる文字は，「カウンタ名」と同じく，ラテンアルファベットの「a-z」「A-Z」，数字「0-9」それにハイフン「-」のみです．はじめの1文字は，ラテンアルファベットのみです．厳密には，ISO 10646 charactersの161番以上も許されます．

```
@page{size: portrait}
@page "land"{size: landscape}
TABLE {page: "land"; page-break-before: right}
```

※ page　ページ名の指定

値	<ページ名> | auto
初期値	auto
適用対象	ブロック系要素
継承される?	yes

上で説明した<ページ名>を指示するためのプロパティです．

7-4 フォント選択の拡張

これは，とくにプロフェッショナル向けだと思われますので，詳細はCSS2仕様書に譲り，ここでは考え方を簡単にまとめるだけにとどめます．

7.4.1. 「@font-face」

CSS2では，記述者が念頭においているフォントの各種特徴をスタイルシート中に書き込むためのセクション「@font-face{}」を定義しました．この情報をもとに，WWWブラウザは「各種特徴が似たフォントを探す」「似たフォントを修正して作る」「フォントをダウンロードする」といった対処が可能になります．

```
@font-face {
    font-family: "Robson Celtic";
    src: url(http://site/fonts/rob-celt)
}
H1 { font-family: "Robson Celtic", serif }
```

CSS2仕様書15.3.1の例を引用して，WWWブラウザの処理を紹介します．先のシートの場合，H1には「Robson Celtic」というフォントファミリーが指定されています．CSS2対応のWWWブラウザは，このフォントがシステムに存在しない場合，対応する@font-face記述を探します．この例では，発見した@font-faceにはソースファイルのURLしか記述されていないため，WWWブラウザはキャッシュからそのファイルを探すか，フォントのダウンロードを試みます．それらによるデータの入手に失敗したら，系統名であるserifに適するフォントをシステム内部から選択し，表示します．

7.4.2. 「@font-face」内部の記述

@font-faceの内部に記述するのは，プロパティではなく記述子（Descriptors）と呼ばれるものです．すなわち，記述子を用いて，フォントの性質を記述するのです．

▶ フォント選択のための記述子

font-family，font-style，font-variant，font-weight，font-stretch，font-size

次の「src（ソース）プロパティ」とあわせて利用します．これらの記述子は，「これこれこういう組み合わせのプロパティが指定された時は，このソースで指定したフォントを使って欲しい」といった情報を記述するものです．値のリストアップはカンマ区切りで行います．

・使い方の例

中ゴジック，極太ゴジックの2つのファミリーが利用できるとします．スタイルシート中に「ゴジック」というファミリーを記述した時に，そのweightの100～500を中ゴジックで，600～900を極太ゴジックで表示させたいとします．それを実現するのは，次の記述です．

```
@font-face{
    font-family:"ゴジック";
    font-weight: 100, 200, 300, 400, 500;
    src: local("中ゴジック");
}
@font-face{
    font-family:"ゴジック";
    font-weight: 600, 700, 800, 900;
    src: local("極太ゴジック");
}
```

※ソースの指定 src

値	[<URL> [format(< 文字列 > [, < 文字列 >]*)]	local(< 文字列 >)] [, <URL> [format(< 文字列 > [, < 文字列 >]*)]	local(< 文字列 >)]* /*URL　フォーマット指定	フォントフェース */
初期値	定義されない

想定するフォントファイルのソースを指定します．その記述方法は以下のとおりです．

・format()によるフォーマット指定

記述したURLのフォントデータのフォーマット（形式）を指定します．想定されているフォーマット名は**表7-6**のとおりです．

表 7-6. フォントフォーマット（CSS2 仕様書 15.3.5 より転載）

キーワード	説明	拡張子
"truedoc-pfr"	TrueDoc™ Portable Font Resource	.pfr
"embedded-opentype"	Embedded OpenType	.eot
"type-1"	PostScript™ Type 1	.pfb, .pfa
"truetype"	TrueType	.ttf
"opentype"	OpenType, including TrueType Open	.ttf
"truetype-gx"	TrueType with GX extensions	
"speedo"	Speedo	
"intellifont"	Intellifont	

・local()によるフォントフェース指定

ローカルフォントをフルネームで書いたもの，と説明されています．

・例（CSS2 仕様書 15.3.5 より転載）

```
src: url(http://foo/bar) ;
src: local("BT Century 751 No. 2 Semi Bold Italic") ;
src: url(../fonts/bar) format("truedoc-pfr") ;
src: url(http://cgi-bin/bar?stuff) format("opentype", "intellifont") ;
```

src 記述子の定義をみると，format()を伴わない URL の記述は許されていないのですが，ここには書かれています．

▶ マッチングに用いる記述子

panose-1，stemv，stemh，slope，cap-height，x-height，ascent，descent

具体的なソースや形状を記述するのではなく，外形的な特徴を指定することで，WWW ブラウザに相当するフォントを探させるものです．ここにあげた記述子は綿密な関連を持っているため，幾つかが欠けただけでも（組み合わせによっては）マッチング計算が不可能になります．

・panose-1

TrueTypeFont のマッチング定数．ラテン系独特の定数．

・stemv，stemh

軸（stem）の vertical と horizontal の幅．

・slope

傾斜．ただし，傾斜があるからといって italic 系だとは限らない．

・cap-height，x-height，ascent，descent
読んで字のごとし．

仕様書では独立した区分になっていますが，次の記述子もいっしょに紹介します．

・units-per-em
1文字を表現するのに使われる座標点の数．マッチング計算で用いられる．css2仕様書によると，具体的な値は**表7-7**のとおり．

表7-7．units-per-em の値の例

キーワード	値
Intellifont	25
Type 1	1000
TrueType，TrueType GX，OpenType	2048

▶ フォント修正に用いる記述子

'widths', 'bbox' and 'definition-src'
既存のフォントを修正して望みに合うようにするための記述子．

・widths
unicode の renge と，その文字が占める width を指定
・bbox
文字を表示するのに必要な矩形
・definition-src
@font-face 指定のある別シートの URL を記述

ほかにも「baseline」「centerline」「mathline」「topline」という記述子があり，それぞれ名前が示すとおりですが，筆者不明のため，具体的にどんな値がどのような意味になるのかは不明です．

▶ その他

・unicode-range
UNICODE のどの範囲をカバーするのかを記述します．

7-5 TABLE

仕様書にあるTABLE関連の記述の多くはWWWブラウザの処理方法（WWWブラウザベンダ向け）ですので，これもある程度省略しながら話を進めます．

7.5.1. display プロパティの値と部位の関係

CSS2が想定するのは必ずしもHTML4.0文書とは限りませんが，TABLEモデルはHTML4.0を基本にしています．XMLなどで文書型定義を自作する方は，**表7-8**の位置づけを参考にしてスタイルシートを記述してください．

表7-8. CSS2におけるTABLEモデル（displayプロパティの値）

キーワード	対応するHTMLの要素
table	TABLE
inline-table	TABLE(display:inline)
table-row	TR
table-row-group	TBODY
table-header-group	THEAD
table-footer-group	TFOOT
table-column	COL
table-column-group	COLGROUP
table-cell	TD, TH
table-caption	CAPTION

7.5.2. 列（column）に指定できるプロパティ）

HTMLのTABLEモデルは，行（row）を基本にしています．そのため，スタイルの細かい調整は行（あるいはセル）単位で行います．列（COL要素，COLGROUP要素）に指定できるプロパティは，次のものに限定されています．

border，background，width，visibility

7.5.3. TABLE 専用プロパティ

※ caption-side　キャプションの位置	
値	top I bottom I left I right
初期値	top
適用対象	table-caption
継承される？	yes

CAPTION（テーブルの見出し）の表示位置を決定します．

※ table-layout レイアウト方法の指定	
値	auto I fixed
初期値	auto
適用対象	table，inline-table
継承される？	no

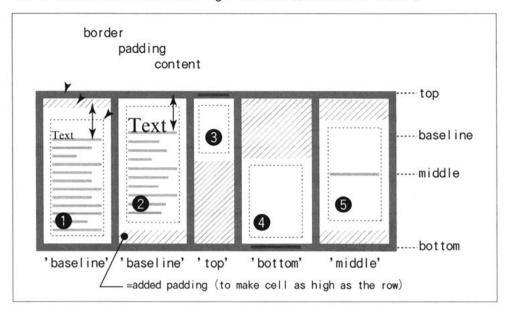
テーブルのレイアウトの指定方法の指定です．autoであれば表示幅にあわせて動的に決定され，fixedであれば表示幅にかかわらず一定サイズで決定されます．処理はfixedの方が速く，またWWWブラウザへの負担が少ないでしょう．しかし，読み手の自由度の面ではautoの方が優れています．

仕様書にはauto時の処理アルゴリズムの指針が書かれていますが，本書では省略します．

7.5.4. table-cell における vertical-align の解釈

TABLEでは，「セルそのもののサイズ」よりも「セルの内容を表示するために必要なサイズ」が小さいことがほとんどです．そのような場合は，自動的にパディングを生成することでつじつまを合わせます．

table-cellにおけるvertical-alignは，セルに内容をレンダリングするときの垂直位置そろえの指定として機能します（**図7-5**）．値として，「baseline」「top」「bottom」「middle」のみが許されます．baselineは「最初の1行のフォントのbaseline」，残りはセルそのものの「上端」「中央」「下端」を意味します．

図 7-5.　table-cell における vertical-align（CSS2 仕様書 17.5.3 より転載）

7.5.5. table-cell における text-align の拡張

text-align の節（5.5.1）で言及したように，「text-align: "."」などとすることで小数点そろえを指示できます．

7.5.6. TABLE のボーダー処理モデルの種類

※ border-collapse　　TABLE のボーダー処理モデルの種類

値	collapse ❘ separate /* 融合　分離 */
初期値	collapse
適用対象	table, inline-table
継承される？	yes

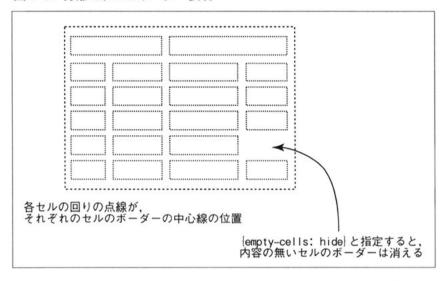
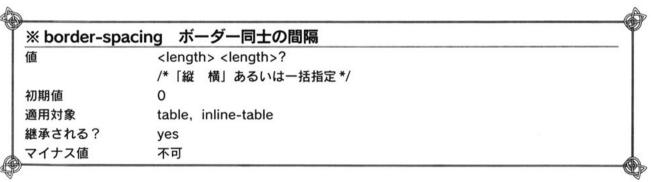
collapse の意味は「倒壊，崩壊，衰弱，折り畳み」なのですが，「collapse モデル」の内容的な意味合いから「融合モデル」という表現を用います．

TABLE のボーダー処理方法を決定します．それぞれの値に応じて，次の項で説明するように処理されます．

7.5.7. 分離（separate）ボーダーモデル

「border-collapse: separate」の場合，セルの1つ1つが独立したボーダーを持つものとして処理されます．すなわち，隣接するボーダーの間には，**さらなる隙間**が存在することになります（**図7-6**）．

図7-6. 分離モデルのボーダー表現

※ border-spacing	ボーダー同士の間隔
値	<length> <length>? /*「縦　横」あるいは一括指定 */
初期値	0
適用対象	table, inline-table
継承される？	yes
マイナス値	不可

セルのボーダー同士の間隔を設定します．セルにボーダーがない場合には，セルとセルの間隔になります（これとは別に，margin や padding も有効ですのでご注意ください）．

このプロパティの定義は，「<length>{1.2}」が正しいような気がします．あくまでも筆者の私見ですが．

※ empty-cells	空セルの処理
値	show ǀ hide
初期値	show
適用対象	table-cell
継承される？	yes

空セルのボーダーを表示するかしないかを指定します（**図7-6**を参照）．これも分離ボーダーモデルのみのプロパティです．

継承の利用による一括指定

empty-cellsプロパティが適用されるのはTD要素やTH要素ですが，シートとしては**TABLE要素**にempty-cellsプロパティを記述しても構いません．TABLE要素にはempty-cellsプロパティは適用されませんが，このプロパティは継承されるため，一括してTD要素やTH要素にempty-cellsプロパティを記述したのと同じ効果が得られます．

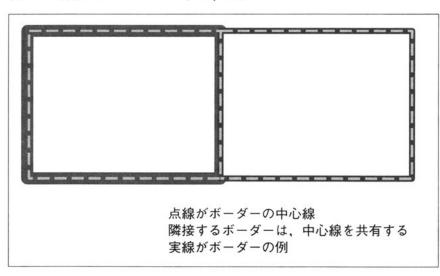
```
TABLE.HAVEEMPTY{empty-cells: hide}
/*HAVEEMPTY クラスの TABLE は，空セルのボーダーは表示しない */
```

同じような効果は，LI要素ではなくUL要素にlist-style系プロパティを指定しても得られます．ちょっとした視点の違いですが，構造を整理するのに役立つ発想だと思われます．

7.5.8. 融合（collapse）ボーダーモデル

▶ 重ねあわせ処理

「border-collapse: collapse」の場合，隣接するセルのボーダーは独立したものではなく，融合したものとして処理されます．正しくは，隣接するボーダーは**同じ中心線をもって**描かれ，後ろのセルのボーダーが前のセルのボーダーの上に描かれていきます．すなわち，隣接するボーダー同士は重なります（**図7-7**）．なお，border-spacingプロパティは無効になります．

図7-7．分離モデルのボーダー表現(一部)

▶ 重ねあわせ処理における「ボーダーなし」処理

ところで,「ボーダーなし」と指定されたセルと「ボーダーあり」と指定されたセルが隣接する場合,隣接部分のボーダーをどのように処理するべきでしょうか. separateモデルでは各々のボーダーは別の位置に描かれますので,どちらの指定も尊重できます.しかし, collapseモデルではボーダーを重ねがきするため,一方を尊重すれば他方の指定を守れないことになります.

この矛盾を処理するために, CSS2ではborder-styleプロパティのキーワードに「hidden」を導入しました.すなわち,「none」は「弱い“なし”」,「hidden」は「強い“なし”」なのです(**表7-9,図7-8**).

表7-9.「ボーダーなし」のコンフリクト処理方法

指定	解説
「boorder-style: none」	隣接部分のborder-styleがnoneかhiddenの場合だけ非表示になります.
「border-style: hidden」	隣接するセルのborder-styleにかかわらず,隣接部分もろとも非表示になります.

上下左右のボーダーで独立して処理されます.この処理は,あくまで融合モデルのみの処理です.

図7-8. TABLEのボーダー非表示処理結果

名前	楽器
John	Guitar & Chor
Paul	Vocal & Base
Ringo	Drums
George	Guitar & Chor

```
TABLE{
   border: solid medium;
}
TH{
   border-style: solid hidden;
   border-width: medium;
}
TD{
   border-style: none hidden;
}
```

TABLE要素には四方にボーダーが指定されているが,
TH, TDともに左右がhiddenであるため,
左右のボーダーは消えている。
TDの上下はnoneであるが,
最初のTD(Johnの行)の上ボーダーはTHによって,
最後のTD(Georgeの行)の下ボーダーはTABLEによって,
それぞれ表示されている。

7-6 ユーザインタフェース

（注：システムフォントについては5.3.3，システムカラーについては5.4.3で言及しました.）

7.6.1. カーソルの指定

※ cursor　カーソル（マウスポインタ）の形状

値	[[<URL> ,]* [auto ❘ crosshair ❘ default ❘ pointer ❘ move ❘ e-resize ❘ ne-resize ❘ nw-resize ❘ n-resize ❘ se-resize ❘ sw-resize ❘ s-resize ❘ w-resize❘ text ❘ wait ❘ help]]
初期値	auto
適用対象	全要素
継承される？	yes

マウス（ポインティングデバイス)を用いるシステム専用のプロパティで，表示された要素の上にカーソル（マウスポインタ）をあわせた時に表示するべきカーソルの形状を指定します.

指定内容は，0個以上のURL（カーソルファイルを指すもの：その後には必ずカンマが必要）と，ひとつのキーワード（必須）です．複数のURLを指定した場合は，前から順に読み込みを試み，最初に成功したものを表示します．ひとつも読み込めなかった場合は，最後の「キーワード」に対応するシステムカーソルを表示します.

カーソルファイルの形式は?

CSS2の仕様書は，カーソルファイルの形式を特定していません．CSS2仕様書の18.1の例では，URLに指定するカーソルの拡張子は「.cur」「.csr」でした.

```
P { cursor : url("mything.cur"), url("second.csr"), text; }
```

しかし，GIFファイルなどが利用されるのではないか，と筆者は考えています．あくまでも私見ですが.

▶ キーワードの解説

auto

WWWブラウザが状況に応じて適切なものを選択します.

crosshair

十字（+）.

default

そのシステムで標準的に用いられるカーソル. 一般的には「矢印」.

pointer

そのWWWブラウザがリンク（アンカー要素）のために利用するカーソルと同じ物.

move

対象が動かせることを示唆するもの.

e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize

サイズ変更が可能であることを示唆するもの. 頭文字は「東西南北」8方位を意味している.

text

選択できるテキストであることを示唆するもの. 一般には「I」.

wait

システムが「忙しい（BUSY）」である旨を示唆するもの. 一般には「時計」「砂時計」.

help

対象にヘルプが用意されていることを示唆するもの. 一般には「?」やバルーン.

7.6.2. アウトラインの指定

▶ アウトラインとは

CSS2が考える「アウトライン（縁取り）」は，ボタンなどのフォーム部品や画像（イメージマップ時は，各部分）の回りに表示される縁取り線です.「ボーダー（枠）」と異なり，アウトラインの形状は対象物の形状と同じ形をしています. すなわち，対象物が丸ければアウトラインも丸いのです（ボーダーの形状は，要素内容の表示に必要な“四角”に固定されています）. あくまで**縁取り線**ですので，ボーダーのような「上下左右」の概念はありません.

「アウトライン」が威力を発揮するのは，「:focus」などのダイナミック疑似要素との連携時でしょう. 普段はアウトラインを表示せず，「マウスをかざした」「選択された」などのアクションに応じてアウトラインを表示するようにすれば，ユーザは「自分が何をしているのか」を明確に意識できます. このような指定は，とくにイメージマップで有効でしょう. なお，この処理を考慮してか，アウトラインの表示／非表示はレイアウトに影響を及ぼさないとされています.

なお，ボーダーを描いている場合，アウトラインはボーダーの外側に描かれます.

※ outline-width	アウトラインの幅
値	[thin	medium	thick	< 一般サイズ >]
初期値	medium
適用対象	全要素
継承される？	no

※ outline-style	アウトラインの線の形状
値	[none	dotted	dashed	solid	double	groove	ridge	inset	outset]
初期値	none
適用対象	全要素
継承される？	no

※ outline-color	アウトラインの色
値	< 色 >	invert /* 任意の色	地の色を反転 */
初期値	invert
適用対象	全要素
継承される？	no

※ outline	アウトライン系の一括指定
値	[<outline-color> || <outline-style> || <outline-width>]
初期値	個々のプロパティの初期値
適用対象	全要素
継承される？	no

アウトライン系のプロパティは，ほぼボーダー系のプロパティと同じです．ひとつだけ違うのが，色指定における「invert：反転」でしょう．これは特定の色を指定するキーワードではなく，対象の色を反転させた色を指定するキーワードです．

7.6.3. 拡大縮小

CSS2仕様書18.5に「Magnification」というものがありますが，ここには「CSS2対応のWWWブラウザであれば，文書表示の拡大縮小機能をサポートしてほしい」とだけ書かれています．とくに拡大縮小のためのプロパティを定義するものではありません．

7-7 音声再生

7.7.1. 音声再生の意義

文書の音声再生というと，一般には視覚障碍者むけの処理だと思われがちですが，実際には皆の役に立つものです．たとえば，**劇の台本**における役者への指示書きとしてCSS2を用いれば，その内容をコンピュータによって客観的に確かめられます（もちろん，芸術性には欠けるでしょうが）．また，CSS2の音声再生の範疇にはBGMや「章の開始音」なども含まれていますので，スタイルシート中に「スクリーン表示用」と「音声再生用」の両方を記述しておけば，それだけで**マルチメディアプレゼンテーション**が可能になります．ほかにも，いわいる**「ながら」仕事**のためには，コンピュータが文書を読み上げてくれると便利だと思いませんか?

```
@media aural{
    P.heidi{/*ハイジ*/
        voice-family: female;/*女の子*/
        pitch: high;/*やや甲高い*/
        pitch-range: 80;/*やや感情の入れすぎ*/
        azimuth: center-left;/*左寄りの位置で再生*/
    }
    P.peter{/*ペーター*/
        voice-family: male;/*男の子*/
        pitch: medium;/*標準的な口調*/
        pitch-range: 50;/*標準的な感情表現*/
        azimuth: center-right;/*右寄りの位置で再生*/
    }
    P.goat{/*ヤギ*/
        play-during: url(meeeee.au) mixed repeat; /*メエエエエ*/
    }
}
```

7.7.2. 音素調整に関するプロパティ

※ voice-family	ボイスファミリー
値	[[< ファミリー名 >	< 系統名 >],]* [< ファミリー名 >	< 系統名 >]
初期値	実装依存
適用対象	全要素
継承される？	yes

フォントファミリーならぬボイスファミリーです．「系統名」には，「male（男性）」「female（女性）」「child（子供）」の3つだけが用意されています．

フォントファミリーファイルならぬボイスファミリーファイルというものは存在するのでしょうか？　フォントファミリーと同様に考えるのであれば，すべての音素の発音が，ある程度以上の音程に対して用意されていなければなりません．「勝新太郎の声」や「オーソン・ウェルズの声」が発売されていたら面白いと思うのですが，現実的にはどうなんでしょう．

※ volume	音量
値	< 数値 >	< パーセント >	silent	x-soft	soft	medium	loud	x-loud /*0 から 100，ただし 0 は無音ではない */ /*silent は無音 */ /*x-soft から順に 0，25，50，75，100 と同じ */
初期値	medium
適用対象	全要素
パーセントの意味	親要素の volume に対する割合
継承される？	yes
マイナス値	不可

音量を0から100の範囲で指定します．ただし，**0 は必ずしも無音ではありません．** 無音はキーワード「silent」で指定します．なお，silent指定では，本来読み上げるのに必要だった時間だけ「休む」ことになります．読み飛ばすのであれば，「display:none」あるいは「speak:none」を指定します．

ダイナミックレンジ

音楽業界の用語で，再生できる最小の音量と最大の音量の間のレンジ（幅）をダイナミックレンジといいます．volume プロパティの 0 ～ 100 は，基本的にはダイナミックレンジに対するパーセントだと解釈できます．

さて，CSS2 の仕様書には，ダイナミックレンジの利用法に関して面白い記述がありますので，要約してみます．

> ・*WWW ブラウザは，状況によって，ダイナミックレンジの一部に volume 指定の 0 から 100 を割り当てるようにしてほしい．たとえば，夜中で近所に迷惑がかかるような状況での再生であれば，volume 指定の 0 ～ 100 を本来のレンジの 0 ～ 40 に割り振ってしまってほしい．*

※ speech-rate	読み上げ速度
値	< 数値 > | x-slow | slow | medium | fast | x-fast | faster | slower /* 順に 80，120，200，300，500，+40，-40*/
初期値	medium
適用対象	全要素
継承される？	yes
マイナス値	不可

読み上げ速度を，1 分間に読み上げる単語数で指定します．コメントにあげた指定は，CSS2 仕様書が提示する「目安」の値です．なお，相対指定キーワード「faster」「slower」は，親要素の指定からの相対指定になります．

※ pitch	ピッチ（声の調子，音程）
値	< 振動数 > | x-low | low | medium | high | x-high /* 低い～高い */
初期値	medium
適用対象	全要素
継承される？	yes
マイナス値	不可

読み上げの平均的なピッチ（声の調子，音程）を指定します．＜振動数＞の単位には「Hz（ヘルツ）」および「kHz（キロヘルツ）」が用意されており，数値は正の実数です．なお，CSS2 仕様書によれば，人間の喋り声の平均的なピッチは，男性で 120Hz，女性で 210Hz です．

用意されているキーワードは，その要素のボイスファミリーにおける標準的な「低い」～「高い」を要求するためのキーワードです．

※ pitch-range ピッチ幅（抑揚具合）

値	< 数値 > /*0 ～ 100*/
初期値	50
適用対象	全要素
継承される？	yes

読み上げにおけるピッチの上下変動具合を指定します．いいかえると，読み上げの抑揚具合を指定します．あまりよいたとえではないかもしれませんが，0だと「棒読み」で，100だと「感情の入れすぎ」になります．

※ stress アクセントの強度

値	< 数値 > /*0 ～ 100*/
初期値	50
適用対象	全要素
継承される？	yes

アクセント位置における音素の発音の強調具合を指定します．数値が大きいほど「おおげさ」になります（指定するのは具体的なHz幅ではなく，相対的な意味合いの0～100の数値です）．

※ richness 音色の豊かさ，明るさ

値	< 数値 > /*0 ～ 100*/
初期値	50
適用対象	全要素
継承される？	yes

仕様書の表現では，「数値が大きいほど“carry”，小さいほど“soft”」となっています．筆者の不明により，よい訳語が見当たりません．このプロパティの値が大きいほど倍音成分が多い音色，すなわち俗にいう「太い音になる」といえば通じるでしょうか．

7.7.3. 読み上げの彩りに関するプロパティ

※ pause-before，pause-after　読み上げの一時停止	
値	< 時間 > I < パーセント >
初期値	実装依存
適用対象	全要素
パーセントの意味	その要素の speach-rate プロパティの逆数に対する割合
継承される？	no
マイナス値	不可

※ pause　読み上げの一時停止の一括指定	
値	[< 時間 > I < パーセント >]{1,2}
初期値	実装依存
適用対象	全要素
パーセントの意味	その要素の speach-rate プロパティの逆数に対する割合
継承される？	no
マイナス値	不可

その単語を読み上げる前後の「一時停止」時間を指定します．一括指定の場合，ひとつだけ指定すれば「前後一括」，2つ記述した場合は順に「前」「後」です．

なお，＜時間＞の単位は「s（秒）」と「ms（ミリ秒）」の2つだけが規定されています．

※ cue-before，cue-after　キュー（音のアイコン）	
値	<URL> I none
初期値	none
適用対象	全要素
継承される？	no

その要素を読み上げる前後に，音の装飾を挿入する場合に指定します．CSS2仕様書では「音のアイコン」と表現されています．なお，音ファイルの形式はWWWブラウザに一任されます．

※ cue　キュー（音のアイコン）の一括指定	
値	<cue-before> II <cue-after>
初期値	個々のプロパティの初期値/* つまり，none*/
適用対象	全要素
継承される？	no

定義と仕様の記述が食い違っています．説明内容が正しいとすれば，ここは「<URL>{1,2}」と書かれているべきです．

キューの一括指定で，ひとつだけ書いた場合は前後両方を，2つ書いた場合は順に「前」「後」を指定したことになります．

※ play-during　BGMとその再生方法の指定

値	[<URL> mix? repeat?] | auto | none
初期値	auto
適用対象	全要素
継承される？	no

要素を読み上げている間に鳴っているサウンド（BGM）の制御のためのプロパティです．

基本的には，URLで指定されたファイルがその要素のBGMになります．キーワード「repeat」があれば繰り返し再生しますが，なければ1度しか再生しません．要素を読み上げ終わると，BGMも鳴り止みます．

HTML文書では，要素は入れ子になっています．そのため，必然的に「親要素の背景」は「子要素の背景」となり，その重ねあわせに関して注意が必要になります．背景画像や背景色の指定では「transparent（透明）」というキーワードがありましたが，ここでは「mix」「auto」「none」の3つのキーワードが用意されており，事情はなかなかに複雑です．

URLを指定した場合には，「mix」キーワードを添えるかどうかを決めなければなりません．親要素のBGMと子要素のBGMの両方を鳴らすのであれば「mix」を指定します．自分のBGMだけを鳴らして親要素のBGMをストップさせるのであれば，URLだけを記述します．後者の場合，その要素が終了したら，親要素のBGMは再開されます．

URLを指定しない場合，すなわちその要素に特にBGMを指定しない場合でも，「auto」か「none」を選択しなければなりません．親要素にBGMがあった場合にそれを受け入れるならば，「auto」を指定します．完全にBGMをなくしたい場合は「none」です．後者の場合でも，その要素が終了したら，親要素のBGMは再開されます．

以上を**表7-10**としてまとめます．

表7-10．play-duringプロパティの指定の解説

BGMを指定する場合のキーワードは，オプション指定	
repeat	繰り返し再生するか否か
mixed	親要素のBGMを再生するか否か

BGMを指定しない場合のキーワードは，排他指定	
auto	親要素のBGMを引き継ぐ，親にBGMが無ければ「無し」
none	親要素にBGMがあっても，BGMを鳴らさない

なお，指定が継承されないのは，「子要素のところで，もういちどBGMを始めから再生し直したりはしない」ためです（CSS2の汎用キーワード「inherit」で明示的に継承した場合は，どうなるのでしょう?）．

7.7.4. 音場における再生位置に関するプロパティ

▶PAN調整とは

ステレオ／サラウンドサウンドでは，同じ音を左右のスピーカーから音量（やタイミング）を変えて再生させることで，その音が知覚される位置を調整できます．そのような調整を「PAN調整」といいます．azimuthプロパティは，左右のPAN調整を指定するプロパティです．

CSS2仕様書では「PAN調整」という用語は使われていません．これは音楽業界のスラングです．

ラジオドラマなどでは，登場人物ごとにPAN位置を変えることで人物の違いを把握しやすいように処理しています．スタイルシートでも同じような指定が可能です．なお，劇などの場合，舞台をイメージする意味で，人物の移動にあわせてPAN位置を変更するケースもあります．このような動的な表現はCSS2では不可能だと考えた方が良いでしょう（SPAN要素を複雑に用いれば，実現できなくも無いですが…）．

※ azimuth　水平のPAN調整

値	< 角度 > | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] || behind] | leftwards | rightwards /* 中央が0度，左がマイナス，右がプラス */
初期値	center
適用対象	全要素
継承される？	yes
マイナス値	可

＜角度＞の単位は「deg（度）」「grad（グラッド）」「rad（ラジアン）」の3つだけが用意されています（**表7-11**）．本書の例ではdeg（度）を用います．

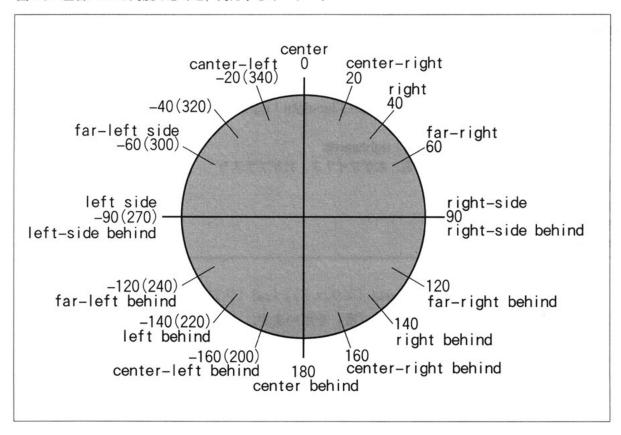
表 7-11．角度の単位

単位	読み方	範囲
deg	度	0度～360度
grad	グラッド	0～100～0～-100～0(90度ごとに)
rad	ラジアン	0～2 π

CSS2にはπに相当する定数は用意されておらず，しかも計算式も記述できないため，各自であらかじめ3.14などをかけてください．

左右のPAN調整では，全円を360度で表現します．正面をゼロとして，**時計回りがプラス**になります．360度=0度，180度がちょうど背面になります．なお，仕様としては「-360度～360度」が記述できるため，「-180度～180度」を用いるのが理解しやすいでしょう（-90度＝270度です）．また，プラス記号「+」は書いても書かなくても同じになります．

用意されているキーワードには，絶対指定と相対指定があります．まず，絶対指定ですが，キーワード「left-size～right-side」とオプションキーワード「behinde」が存在します．behinde無しでは前面（-90度～90度）が表現でき，behindeを付けると背面（90度～180度，-180度～-90度）が表現できます．数値による絶対指定とあわせて，**図7-9**に示します．

図7-9．左右PANの角度の意味と，対応するキーワード

相対指定は，親要素の再生位置からの相対指定になります．時計回り方向（プラス）が「rightward」，その逆（マイナス）が「leftward」で，どちらも20度ずつの移動になります．なお，「値の限界を超えたらどうなるか」には言及されていませんが，回転しつづけると解釈してかまわないでしょう．

「behind」単独指定について

定義としては，キーワード「behind」の単独指定も可能です．しかし，仕様書にはその場合の処理が記述されていません．可能性としては，相対指定であるとして「鏡あわせに移動」「180度移動」が考えられます．

しかし，仕様書中の指定例では，「`P.comment { azimuth: behind }  /* 180deg */`」とあるので，ひょっとすると「`center behind`」として扱うのかもしれません．

個人的には，これも定義ミスで，本来は次のようになっているべきなのではないか，と考えています．

※ azimuth　水平の PAN 調整

```
[
  [ left-side | far-left | left | center-left | center |
  center-right | right | far-right | right-side ]
  behind ?
] | leftwards | rightwards
/* 中央が 0 度，左がマイナス，右がプラス */
```

※ elevation　垂直の PAN 調整

値	< 角度 > | below | level | above | higher | lower /* 水平が 0 度，下がマイナス，上がプラス */ /*-90deg，0deg，90deg，+10deg，-10deg*/
初期値	level
適用対象	全要素
継承される？	yes
マイナス値	可

上下のPAN調整では，-90度～90度を用います．キーワードの意味はコメントのとおりです．なお，相対指定キーワード「higher」「lower」は，親要素の指定からの相対指定になります．

7.7.5. その他

※ speak　喋り方の根本

値	normal | none | spell-out
初期値	normal
適用対象	全要素
継承される？	yes

「display」プロパティの音声再生専用の拡張だと捉えてください（displayプロパティは，すべてのメディアを対象にしたプロパティです）．といっても，CSS2の段階では「普通に読むか，スペルで読むか」しか指定できません．

「スペルで読む」とは，英語圏の場合，spellという単語を「スペル」と読むのではなく，「s, p, e, l, l（エス，ピー，イー，エル，エル）」と読む，ということです．具体的には，「ODA」を「おだ」ではなく「オー，ディー，エイ」と読んでほしいならば，「speak: spell-out」を指定します．

「speak: none」は「読み飛ばす」ための指定です．「display:none」は子要素もろとも読み飛ばしますが，「speak:none」では子要素は読み飛ばしません．

CSS2仕様書には確かにこのように書かれているのですが，定義と説明が逆ではないか，と思われます．displayプロパティは継承されないため子要素には影響を与えず，speakプロパティは継承されるため子要素にも影響が出るのではないでしょうか．

HTML4.0のABBR要素とACRONYM要素

先の「ODA」を「<SPAN CLASS="RYAKUGO">ODA</SPAN>」などと表現しても構いませんが，実は，HTML4.0の新しいインライン要素として，「頭文字・略語」に相当する「ABBR要素」と「ACRONYM要素」が用意されています．こちらをもちいるのが本道でしょう．

```
ABBR, ACRONYM{speak: spell-out}
```

ところが，日本人にはこのふたつの要素の使い分けが良く分からないのです(ToT)

なお，HTML4.0仕様書9.2.1の例では，ABBR要素に適した単語として「WWW，HTTP，URI，Mass」があげられており，ACRONYM要素として「WAC，Rader」があげられています．

※ speak-punctuation 記号の読み上げ方

値	code I none
初期値	none
適用対象	全要素
継承される？	yes

記号「{}()[];:」などの読み方を指定します．通常は「none」，すなわち，読み上げません．「speak-punctuation: code」と指定すると，「カッコ開く」などと読み上げます．

※ speak-numeral　数値の読み上げ方

値	digits I continuous
初期値	continuous
適用対象	全要素
継承される？	yes

数値の読み上げかたを指定します．たとえば，「123」の場合，「continuous」では「ひゃくにじゅういち」などと読み，「digits」では「いちにいさん」と読みます．

7.7.6. テーブルの読み上げ

▶「データのセル」と「見出しのセル」の関係の表現方法

HTMLのTABLE要素で作成できる表の構成要素には，大きく分けて「データのセル（TD要素）」と「見出しのセル（TH要素）」があります．ところで，「あるセルの見出しを探すときに表を行方向に見るべきか，列方向に見るべきか」は，HTML3.2のマークアップでは判別できません．また，見出しは行列の両方に存在するかもしれません．あるいは，同じ方向に2つ以上の見出しがあるかもしれません．このような場合には，一方だけが意味を持つのでしょうか，双方が関連しているのでしょうか?（**図7-10**）．

図7-10．見出し構造の複雑な表の例

名前		楽器	
名	姓	主	副
ジョン	レノン	ギター	コーラス
ポール	マッカートニー	ボーカル&ベース	
リンゴ	スター	ドラム	コーラス
ジョージ	ハリスン	ギター	コーラス

図7-10を見ても分かるように，HTML3.2の不十分なTABLEマークアップであっても，WWWブラウザでコンピュータ画面に表示してしまえば，見出しの関係はなんとなく判断できます．だからといって，そのままでは情報として不十分です．そこで，HTML4.0では「HEADERS属性」「SCOPE属性」「AXIS属性」を導入して，どの見出しがどのセルを対象にしているのかを記述できるようになりました．いいかえれば，見出しによってセルを分類できるようになったのです．詳しくはHTML4.0仕様書11章を参照願いたいのですが，以降の例を読んだだけでも，ある程度はご理解いただけると思います．

▶ 表の音声再生における「見出しのセル」表現

ところで，皆さんは表を見ることができない人に，表の内容を読み上げることによってその内容を伝えようとしたことがあるでしょうか．無いならば想像してほしいのですが，各セルを左上から順に機械的に読み上げたときに，その内容がどれだけ伝わるでしょうか？　おそらく何も伝わらないと思います．

表の内容を読み上げで表現するには，なんらかの工夫が必要になります．そのためにCSS2が取りいれた工夫は，「データのセルを読み上げるときに，その見出しを読み直す」というものです．

※ speak-header　テーブル見出し読み上げの制御

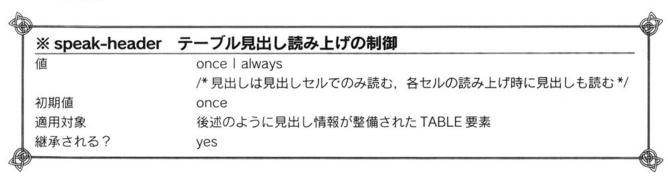
値	once I always /* 見出しは見出しセルでのみ読む，各セルの読み上げ時に見出しも読む */
初期値	once
適用対象	後述のように見出し情報が整備されたTABLE要素
継承される？	yes

「speak-header: always」と指定すると，先に延べたように「データのセルを読み上げる時に，その見出しを読み直す」ことになります．**図7-11**に，読み上げイメージを書き留めます．

図7-11．TABLEの読み上げイメージ

名前	楽器
John	Guitar & Chor
Pau	Vocal & Base
Ringo	Drums
George	Guitar & Chor

→

名前：John　楽器：Guitar & Chor
名前：Paul　楽器：Vocal & Base
名前：Ringo　楽器：Drums
名前：George　楽器：Guitar & Chor

▶ 付録： SCOPE属性の使い方

図7-11 のTABLEの記述例を提示します．

見出しが行あるいは列の一方にしかない場合は，TH要素のSCOPE属性によって，見出し構造の方向を表現できます．属性値のキーワードとしては，さしあたっては「ROW（行）」と「COL（列）」だけ覚えれば十分でしょう．

```
<TABLE SUMMERY="この表は，ビートルズのメンバー名と，一般的な担当楽器を示すものです．">
<CAPTION>ビートルズのメンバー表</CAPTION>
    <TR>
        <TH SCOPE="COL">名前</TH>  <TH SCOPE="COL">楽器</TH>
    </TR>
    <TR>
        <TD>John</TD>  <TD>Guitar&amp; Chor</TD>
    </TR>
    <TR>
        <TD>Paul</TD>  <TD>Vocal &amp; Base</TD>
    </TR>
    <TR>
        <TD>Ringo</TD>  <TD>Drums</TD>
    </TR>
    <TR>
        <TD>George</TD>  <TD>Guitar&amp; Chor</TD>
    </TR>
</TABLE>
```

▶ 付録： HEADRES属性の使い方

見出しが行列両方に存在したり，列だけでも「大見出し」と「小見出し」が存在する場合（**図7-10**）には，HEADRES属性を用います．

図7-10．見出し構造の複雑な表の例（再掲）

名前		**楽器**	
名	姓	主	副
ジョン	レノン	ギター	コーラス
ポール	マッカートニー	ボーカル&ベース	
リンゴ	スター	ドラム	コーラス
ジョージ	ハリスン	ギター	コーラス

このような場合は，TH要素にID属性をつけ，TD要素に対応する見出しセルのID名を記述します．複数の見出しと関係する場合は，大見出しから順に空白で区切って書き上げます．

厳密には，かならずしも「大見出しから順に」と決められているわけではありません．ただ，そのように活用したほうがより効果的です．

```
<TABLE SUMMERY="この表は，ビートルズのメンバー名と，一般的な担当楽器を示すものです．">
<CAPTION>ビートルズのメンバー表</CAPTION>
<TR>
    <TH ID="name" COLSPAN="2">名前</TH>
    <TH ID="inst" COLSPAN="2">楽器</TH>
</TR>
<TR>
    <TH ID="first">名</TH>
    <TH ID="family">姓</TH>
    <TH ID="main">主</TH>
    <TH ID="sub">副</TH>
</TR>
<TR>
    <TH HEADRES="name first">ジョン</TH>
    <TH HEADERS="name family">レノン</TH>
    <TH HEADERS="inst main">ギター</TH>
    <TH HEADERS="inst sub">コーラス</TH>
</TR>
<TR>
    <TH HEADRES="name first">ポール</TH>
    <TH HEADERS="name family">マッカートニー</TH>
    <TH HEADERS="inst main">ボーカル&amp;ベース</TH>
    <TH HEADERS="inst sub"></TH>
</TR>
<TR>
    <TH HEADRES="name first">リンゴ</TH>
    <TH HEADERS="name family">スター</TH>
    <TH HEADERS="inst main">ドラム</TH>
    <TH HEADERS="inst sub">コーラス</TH>
</TR>
<TR>
    <TH HEADRES="name first">ジョージ</TH>
    <TH HEADERS="name family">ハリスン</TH>
    <TH HEADERS="inst main">ギター</TH>
    <TH HEADERS="inst sub">コーラス</TH>
</TR>
</TABLE>
```

「アクセス性」という思想

HTML4.0の仕様において,「アクセス性の確保」を盛り込むことは重大な課題でした.「アクセス性の確保」とは,「マウスを使わなくても,キーボードショートカットだけでフォームやリンクを操作できる」「印刷結果(インタラクティブ性無し)でもリンクなどの意図が伝えられる」「音声再生でも文書を理解できる」といったようなことです.そのために,アンカー要素やフォーム部品には「ACCESSKEY属性」が用意されましたし,ほぼすべての要素にTITLE属性(IMG要素のALT属性と同等)が用意されました.先程紹介したHEADER属性やSCOPE属性は必ずしも音声読み上げを可能にするための属性ではありませんが,これらによって(表の見出し構成を明確にすることで)音声再生でも表を理解可能になります.

しかし,「コンピュータ画面に表示すること」「マウスを用いること」などを前提にしてしまえば,このような属性を用いなくても記述者の意図はなんとなく伝わってしまうものです.そのせいか,暗黙のうちに「コンピュータ画面」「マウス」を前提にするきらいがあり,既存のHTML解説書はアクセス性に関する新しい属性の存在に言及せずに済ましてしまうケースがままあります.お手もとのHTML4.0に関する書籍の中に,ACCESSKEY属性やSCOPE属性の解説があるかどうかチェックしてみてください.たぶん,記述されていないと思います.本書のHTML解説部分でも省略しています.これは大変に遺憾なことです.

画像を多用される方,TABLEを多用される方,フォームを多用される方,フレームを多用される方は,ぜひ一度アクセス性の確保についてご考慮ください.その際には,W3Cの内部機関WAI(Web Accessibility Initiative)の文書を参考にするとよいでしょう.

http://www.w3.org/WAI/

資料

a. スタイルシートの先方互換（不明な記述の処理方法）

レベル2のCSSをサポートするWWWブラウザが，レベル1のスタイルシートを読みこなせるのは普通のことです．これを後方互換（backward-compatibility）と呼びます．しかし，レベル1しかサポートしていないWWWブラウザがレベル2以上のスタイルシートを読み取った場合には，当然，一部の指定が処理できないはずです．そのような際に（すべてを無効にするのではなく）一部だけでも有効にするための「不明な記述の処理方法」が定められていますので，ここで簡単に紹介します．W3Cはこれを「先方互換（forward-compatibility）」と呼んでいます．

◆知らない＠ルールは無視する

受け取ったシート：

```
@three-dee {  /*まったく知らない@ルール*/
      @import "basic.css";
      H1 { color: red }
}
H1 {color: blue}
```

↓

処理後：

```
H1 {color: blue}
```

◆知らないプロパティのある宣言は無視する

受け取ったシート：

```
H1{
      color: white;
      background: block;
      text-shadow: #888 #444 #222;  /*CSS2のプロパティ*/
}
```

↓

処理後：

```
H1{
      color: white;
      background: block;
}
```

◆知らない値のある宣言は無視する

受け取ったシート：

```
P IMG{
        float: left top; /*CSS1にはtopというキーワードはない*/
        height: 1.5em:
}
```

↓

処理後：

```
P IMG{
        height: 1.5em:
}
```

b. 現在のWWWブラウザのCSS1サポート状況

本書執筆時では，各ブラウザはCSS2成立以前にリリースされたものであるため，今回の調査はCSS1に限定しました．また，この調査はWindows 95システムにおけるInternet Explorer4.0日本語版（WWWブラウザのみの設定）とNetscape Navigator4.0日本語版のみに限定してあります．マッキントッシュや各種UNIXをお使いの方，あるいはほかのWWWブラウザを用いている方は，あらかじめその旨をご了承くださるようお願い申し上げます．WWWブラウザのバージョンは，Netscape Navigator4.04，Internet Explorer4.00です．執筆段階での最新バージョンはNetscape Navigator4.05，Internet Explorer4.01ですが，これらはマイナーチェンジ（であるはず）なので，大きな違いはないものとさせていただきます．

調査した内容は以下のとおりです．ただし，実際にシートを構築していく上で別途気づいた点も報告します．なお，「正しく機能するのが当たり前」と考え，正常である場合は省略します．

◆基本チェック

・各プロパティがそれぞれ独立して動作するか（無効｜有効だがおかしい）

これを「ブロック系要素（実際にはH1，H2，P，BLOCKQUOTE，DIV）」「インライン系要素（STRONG，A，SPAN）」「リスト（UL，OL，LI）」「置換要素（IMG）」について調べた．

◆応用チェック

・emユニットは正しく処理されるか
・インライン系要素にボーダーラインはつくか

・マイナス値を受け付けるか
・ボーダー表示の入れ子は可能か
・ボーダー内部に表示されるときも，各種指定は正常か
・継承は正しく処理されるか
・カスケードは正しく処理されるか（含む：ユーザ標準シート）
・シート選択，MEDIA選択は機能するか（→4章で報告したとおり，どちらも無効だった）
・シートをoffにできるか
・相対URL計算は正しいか
・印刷できるか
・その他，不条理な処理ミスをしていないか

(1) Internet Explorer4.0

全体の評価

本書では触れませんでしたが，CSS1の仕様書はプロパティを“積極的にサポートするべき”コア・プロパティとそうでないものに分け，WWWブラウザの開発者に対して「コア・プロパティは必ず守ってほしいが，そうでないものは省略してもよい」という指針を与えています．

Internet Explorer4.0は，この「コア」に関してはほぼサポートしていますが，それ以外のサポートはあまり進んでいません．誤解をおそれずにまとめれば，「派手な部分は省略されているが，堅実」だといえるでしょう．「有効になるものは，常に正常」であり，ずいぶん「真面目」だといえます．

しかし，displayプロパティの指定が無効になるなど，複雑なレイアウトを実現するためのいくつかの機能がサポートされていないのは残念なことです（もっとも，CSS1仕様書はこれらをコア・プロパティに含めていません）．

常に無効になるプロパティ

◆全体

・スタイルシートを無効にできない．
・@importは「url(~)」形式しか受けつけない．
・スタイルシートの“選択”機能がない．
・疑似要素（first-line，first-letter）は無効．
・エラー処理が甘いのか，キーワードに""を付けても有効になる．

◆フォント系

・font-variantのsmall-capsはたんなる「大文字」になる．

◆色と背景効果系

すべて有効

◆**テキスト属性系**

- word-spacingは無効（letter-spacingは有効）.
- vertical-alignはsuper, sub以外は無効.
- text-decorationのblinkが無効.
- text-alignのjustifyは複数行にわたるブロックにのみ有効.

◆**ボックス系，表示位置調整**

- インライン系要素にマージン，パディング，ボーダーラインを付けても無効になる（SPAN要素でwidthが明示されている場合は，有効にはなるが表示結果が正しくない）.
- border-styleのdottedとdashedが無効（solidと解釈される）.
- DIV, SPANを除く文字セクションのwidth, heightが無効になる（画像などはOK）.
- DIV, SPANを除く文字セクションのフロート化が無効になる.

◆**その他の表示調整系**

- displayプロパティの「変更」が無視される．noneは有効.
- white-spaceの変更が無効.

(2) Netscape Navigator4.0

全体の評価

Internet Explorer4.0とは対照的に，「派手な部分のサポートが進んでいるが，土台部分の不具合が多い」とまとめられるでしょう．とくにemユニットの不完全な処理は，ときにはレイアウトの致命的な崩壊をもたらします．しかし，不具合を把握し，それを避けるように注意しながらスタイルシートを記述すれば，効果的なレイアウトを実現できると言えます.

設計は「理想図」に基づいて行う

ここでとくにアピールしておきたいことは，「Netscape Navigator4.0の表示結果だけに基づいてスタイルシートを記述すると，おかしなシートになってしまう」という点です.

“フォントサイズを「親要素の3倍」にしたいのに，3emと記述したら6倍くらいになってしまった．じゃあ，1.5emと書いておけばちょうどいいや”

などと考えてしまうと，1.5emを「正しく」解釈するWWWブラウザでは望みどおりの効果は得られません.

スタイルシートの設計は，あくまで“理想図”に基づいて行ってください．そうしなければ，スタイル調整の汎用性は得られません.

理想的なシートを記述した上でならば，現在のWWWブラウザの不具合にあわせて，たとえば部分的にpt（ポイント）で記述し直すなどで対応します.

常に無効になるプロパティ

◆全体

・疑似要素（first-line，first-letter）は無効．

・疑似クラスのactiveが無効．

・@importが無効（相当するLINK要素をHTML文書に記述することで対処できる）．

・スタイルシートの“選択”機能が無い

・ユーザ標準スタイルシートを指定できない．

・エラー処理が一部不正（指定値が不正だと，{}内がすべて無効になる）．

・印刷結果が画面表示と大きく異なる．

◆フォント系

・font-variantのsmall-caps，font-styleのobliqueが無効．

◆色と背景効果系

・background-position，background-attachmentが無効．

◆テキスト属性系

・text-alignのjustify，text-decorationのoverlineが無効．

・vertical-alignは完全に無効．

・word-spacing，letter-spacingは無効．

・text-transformは文字コードが英語ならば有効だが，日本語コードでは無効．

◆ボックス系，表示位置調整

・border-top，border-right，border-bottom，border-leftプロパティが無効（そのため，本書では常に「border-width，border-style，border-color」を利用しました）．

・heightが無効．

・置換要素にはwidth，heightともに無効．

・border-widthの%指定が有効になる（本来は“エラー”）．

◆その他の表示調整系

・white-spaceのnowrapが無効（normalと解釈される）．

・list-style-imageが無効．

・list-style-positionが無効（常にoutside）．

有効だが処理がおかしいもの

・相対URLが，スタイルシートからでなくそれを読み込んだHTML文書から計算される．

・emユニットの計算がおかしい．「その要素のフォントサイズ」であるべきなのに，「親要素のフォントサイズ」として処理されたり，そのほかの怪しい結果になる場合がある．

・line-heightの計算がおかしい．その継承処理も不正．

・背景に画像を指定すると，背景色がtransparentされなくなる．

・下マージンを設定すると，上マージンもその値になる．

・インライン系要素に指定した背景色が正しく表示されない．ブロック系要素では正常．
・インライン系要素にボーダーラインを設定すると，勝手にdisplayプロパティがblockに変更される．あえてdisplay：inlineと明示するとインライン表示されるが，今度はwhite-spaceがnowrapに固定される．
・ボーダーラインをつけると，それまで有効だったfont設定などが無効になったり，表示サイズが変動するなどの不具合を起こす場合がある．
・ブロック系要素の上下ボーダーラインの長さが「文字を表示するのに必要な幅」になる場合がある．左右マージンを指定していれば正常に動作する．
・カスケーディング処理が一部不正．たとえば，BLOCKQUOTE要素やリストにマージンを指定すると，WWWブラウザ標準のマージンに「新たなマージンを追加」することになる．
・LI，HR要素に対する指定の解釈がおかしい．たとえば，LI要素にボーダーラインを付けると，行頭マークだけについてしまう．本来は，項目の語句と行頭マークを囲むボーダーラインが必要．
・HTML文書中で「終了タグを省略していい場合」でも，終了タグを省略していると動作に不具合が現れる場合がある．詳しい組み合わせははっきり言えないが，DIV要素を複雑に使用している場合，P要素の終了タグを省略するとおかしくなる可能性が高かった．
・シートが長すぎたり，文書が長すぎたりすると暴走する．
・そのほか，原因不明で指定が無効にされる場合がある．

Netscape Navigator4.0はCSSをサポートしていない?

じつは，Netscape社は独自仕様のスタイルシートとしてJavascript Stylesheet（コンテントタイプは"text/javascript"）を提唱しています．

同社がWWWで公開している資料によれば，Netscape Navigatorはバージョン4.0からCSSをフルサポートしていることになっていますが，同社が示すリファレンスで解説されている文法やプロパティは一部CSSとは異なるものです．さらに，Javascript Stylesheetと仕様が混同している部分がいくつも見られます．

```
http://home.netscape.com/
http://developer.netscape.com/library/documentation/communicator/dynhtml/index.htm
```

textAlignプロパティなどというモノが存在したり，スタイルシートの拡張子が「.htm」だったりと，なかなか不思議な世界が待っています^^;

また，Netscape Navigator4.0の設定で「スタイルシートを有効」にしていても，「Javascriptを無効」にするとスタイルシートが機能しません．

以上のことから，「実はNetscape Navigator4.0はまったくCSSをサポートしておらず，単に内部的にJavascript Stylesheetに変換しているだけではないか，しかも変換はうまく機能していないのではないか」と想像できます．そうだからこそ，ここまでCSSの仕様と照らし合せた場合に不具合が生じているのではないでしょうか．

C. CSS2のメディア区分リファレンス

◆メディア区分

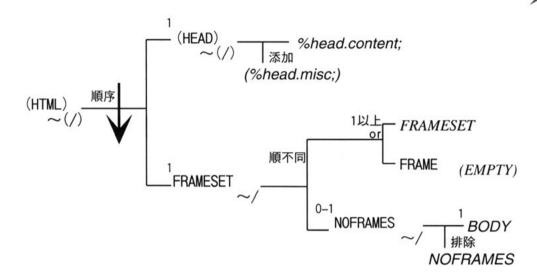
着眼点	区分
大区分	画像(visual), 音声(aural), 接触(tactile ：点字など)
ページ区切り	ある(continuous), ない(paged)
画素	グリッド(grid ：文字端末など), ビットマップ(bitmap)
対話性	ある(interactive), ない(static ：印刷物など)

◆メディアタイプが意味するメディアの性質

メディアタイプ名	大区分	ページ区切り	画素	対話性
aural	aural	continuous	なし	both
braille	tactile	continuous	grid	both
emboss	tactile	paged	grid	both
handheld	visual	both	both	both
print	visual	paged	bitmap	static
projection	visual	paged	bitmap	static
screen	visual	continuous	bitmap	both
tty	visual	continuous	grid	both
tv	visual, aural	both	bitmap	both

d. HTML4.0クイックリファレンス

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">に上書き

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
との差分情報付き

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
%旧来からのblock要素;
or
P ～(/)
0以上 %inline;
%preformatted;
PRE ～/
0以上 %inline;
排除 (%pre.exclusion;)
≡ (IMG | OBJECT | APPLET | BIG | SMALL | SUB | SUP | FONT | BASEFONT)
APPLET, FONT, BASEFONTは定義されません
DL ～/
1以上 or
DT ～(/)
0以上 %inline;
DD ～(/)
0以上 %flow;
%list;
or
UL ～/
OL ～/
1以上 LI ～(/)
0以上 %flow;
DIR ～/
MENU ～/
1以上 LI
排除 (%block;)
DIR, MENUは削除されます
DIV ～/
CENTER ～/
0以上 %flow;
CENTERは削除されます
BLOCKQUOTE ～/
0以上 %flow;
0以上の%flow;ではなく, 1以上の (%block; | SCRIPT) になります
HR (EMPTY)
TABLE ～/
順序
0-1 CAPTION ～/
0以上 %inline;
or
0以上 COL (EMPTY)
0以上 COLGROUP ～(/)
0以上 COL
0-1 THEAD ～(/)
0-1 TFOOT ～(/)
1以上 (TBODY) ～(/)
1以上 TR ～(/)
1以上 or
TH ～(/)
TD ～(/)
0以上 %flow;
FORM ～/
0以上 %flow;
排除 (FORM)
0以上の%flow;ではなく, 1以上の (%block; | SCRIPT) になります
ISINDEX
ISINDEXは削除されます

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

e. CSS2クイックリファレンス

音声再生 ☞p.227

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>azimuth</td><td>< 角度 > | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] || behind] | leftwards | rightwards | inherit</td><td>aural</td></tr>
<tr><td>cue</td><td>['cue-before' || 'cue-after'] | inherit</td><td>aural</td></tr>
<tr><td>cue-before, cue-after'</td><td><URL> | none | inherit</td><td>aural</td></tr>
<tr><td>elevation</td><td>< 角度 > | below | level | above | higher | lower | inherit</td><td>aural</td></tr>
<tr><td>pause</td><td>[[< 時間 > | < パーセント >]{1,2}] | inherit</td><td>aural</td></tr>
<tr><td>pause-before, pause-after</td><td>< 時間 > | < パーセント > | inherit</td><td>aural</td></tr>
<tr><td>pitch</td><td>< 振動数 > | x-low | low | medium | high | x-high | inherit</td><td>aural</td></tr>
<tr><td>pitch-range</td><td>< 数値 > | inherit</td><td>aural</td></tr>
<tr><td>play-during</td><td><URL> mix? repeat? | auto | none | inherit</td><td>aural</td></tr>
<tr><td>richness</td><td>< 数値 > | inherit</td><td>aural</td></tr>
<tr><td>speak</td><td>normal | none | spell-out | inherit</td><td>aural</td></tr>
<tr><td>speak-header</td><td>once | always | inherit</td><td>aural</td></tr>
<tr><td>speak-numeral</td><td>digits | continuous | inherit</td><td>aural</td></tr>
<tr><td>speak-punctuation</td><td>code | none | inherit</td><td>aural</td></tr>
<tr><td>speech-rate</td><td>< 数値 > | x-slow | slow | medium | fast | x-fast | faster | slower | inherit</td><td>aural</td></tr>
<tr><td>stress</td><td>< 数値 > | inherit</td><td>aural</td></tr>
<tr><td>voice-family</td><td>[[< ファミリー名 > | < 系統名 >],]* [< ファミリー名 > | < 系統名 >] | inherit</td><td>aural</td></tr>
<tr><td>volume</td><td>< 数値 > | < パーセント > | silent | x-soft | soft | medium | loud | x-loud | inherit</td><td>aural</td></tr>
</table>

ページ（印刷） ☞p.209

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>marks</td><td>[crop || cross] | none | inherit</td><td>visual, paged</td></tr>
<tr><td>orphans</td><td>< 整数 > | inherit</td><td>visual, paged</td></tr>
<tr><td>page</td><td>< ページ名 > | auto</td><td>visual, paged</td></tr>
<tr><td>page-break-before, page-break-after</td><td>auto | always | avoid | left | right | inherit</td><td>visual, paged</td></tr>
<tr><td>page-break-inside</td><td>avoid | auto | inherit</td><td>visual, paged</td></tr>
<tr><td>size</td><td>< 一般サイズ >{1,2} | auto | portrait | landscape | inherit</td><td>visual, paged</td></tr>
<tr><td>widows</td><td>< 整数 > | inherit</td><td>visual, paged</td></tr>
</table>

TABLE ☞p.218

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>border-collapse</td><td>collapse | separate | inherit</td><td>visual</td></tr>
<tr><td>border-spacing</td><td>< 一般サイズ > < 一般サイズ >? | inherit</td><td>visual</td></tr>
<tr><td>caption-side</td><td>top | bottom | left | right | inherit</td><td>visual</td></tr>
<tr><td>empty-cells</td><td>show | hide | inherit</td><td>visual</td></tr>
<tr><td>table-layout</td><td>auto | fixed | inherit</td><td>visual</td></tr>
</table>

色と背景 ☞p.125

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>background</td><td>['background-color' || 'background-image' || 'background-repeat' || 'background-attachment' || 'background-position'] | inherit</td><td>visual</td></tr>
<tr><td>background-attachment</td><td>scroll | fixed | inherit</td><td>visual</td></tr>
<tr><td>background-color</td><td>< 色 > | transparent | inherit</td><td>visual</td></tr>
<tr><td>background-image</td><td><URL> | none | inherit</td><td>visual</td></tr>
<tr><td>background-position</td><td>[[< パーセント > | < 一般サイズ >]{1,2} | [[top | center | bottom] || [left | center | right]]] | inherit</td><td>visual</td></tr>
<tr><td>background-repeat</td><td>repeat | repeat-x | repeat-y | no-repeat | inherit</td><td>visual</td></tr>
<tr><td>color</td><td>< 色 > | inherit</td><td>visual</td></tr>
</table>

ボックス ☞p.138

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>border</td><td>['border-width' || 'border-style' || < 色 >] | inherit</td><td>visual</td></tr>
<tr><td>border-color</td><td>< 色 >{1,4} | transparent | inherit</td><td>visual</td></tr>
<tr><td>border-style</td><td><border-style>{1,4} | inherit</td><td>visual</td></tr>
<tr><td>border-top, border-right, border-bottom, border-left</td><td>['border-top-width' || 'border-style' || < 色 >] | inherit</td><td>visual</td></tr>
<tr><td>border-top-color, border-right-color, border-bottom-color, border-left-color</td><td>< 色 > | inherit</td><td>visual</td></tr>
<tr><td>border-top-style, border-right-style, border-bottom-style, border-left-style</td><td><border-style> | inherit</td><td>visual</td></tr>
<tr><td>border-top-width, border-right-width, border-bottom-width, border-left-width</td><td><border-width> | inherit</td><td>visual</td></tr>
<tr><td>border-width</td><td><border-width>{1,4} | inherit</td><td>visual</td></tr>
<tr><td>margin</td><td><margin-width>{1,4} | inherit</td><td>visual</td></tr>
<tr><td>margin-top, margin-right, margin-bottom, margin-left</td><td><margin-width> | inherit</td><td>visual</td></tr>
<tr><td>padding</td><td><padding-width>{1,4} | inherit</td><td>visual</td></tr>
<tr><td>padding-top, padding-right, padding-bottom, padding-left</td><td><padding-width> | inherit</td><td>visual</td></tr>
</table>

フォント ☞p.116

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>font</td><td>[['font-style' || 'font-variant' || 'font-weight']? 'font-size' [/ 'line-height']? 'font-family'] | caption | icon | menu | message-box | small-caption | status-bar | inherit</td><td>visual</td></tr>
<tr><td>font-family</td><td>[[< ファミリー名 > | < 系統名 >],]* [< ファミリー名 > | < 系統名 >] | inherit</td><td>visual</td></tr>
<tr><td>font-size</td><td>xx-small | x-small | small | medium | large | x-large | xx-large |larger | smaller | < 一般サイズ > | < パーセント ></td><td>visual</td></tr>
<tr><td>font-size-adjust</td><td>< 数値 > | none | inherit</td><td>visual</td></tr>
<tr><td>font-stretch</td><td>normal | wider | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded | inherit</td><td>visual</td></tr>
<tr><td>font-style</td><td>normal | italic | oblique | inherit</td><td>visual</td></tr>
<tr><td>font-variant</td><td>normal | small-caps | inherit</td><td>visual</td></tr>
<tr><td>font-weight</td><td>normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit</td><td>visual</td></tr>
<tr><td>line-height</td><td>normal | < 数値 > | < 一般サイズ > | < パーセント > | inherit</td><td>visual</td></tr>
</table>

テキスト ☞p.132

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>letter-spacing</td><td>normal | < 一般サイズ > | inherit</td><td>visual</td></tr>
<tr><td>text-align</td><td>left | right | center | justify | < 文字列 > | inherit</td><td>visual</td></tr>
<tr><td>text-decoration</td><td>none | [underline || overline || line-through || blink] | inherit</td><td>visual</td></tr>
<tr><td>text-indent</td><td>< 一般サイズ > | < パーセント > | inherit</td><td>visual</td></tr>
<tr><td>text-shadow</td><td>none | [< 色 > || < 一般サイズ > < 一般サイズ > < 一般サイズ >? ,]* [< 色 > || < 一般サイズ > < 一般サイズ > < 一般サイズ >?] | inherit</td><td>visual</td></tr>
<tr><td>text-transform</td><td>capitalize | uppercase | lowercase | none | inherit</td><td>visual</td></tr>
<tr><td>vertical-align</td><td>baseline | sub | super | top | text-top | middle | bottom | text-bottom | < パーセント > | < 一般サイズ > | inherit</td><td>visual</td></tr>
<tr><td>word-spacing</td><td>normal | < 一般サイズ > | inherit</td><td>visual</td></tr>
</table>

自動生成文字 ☞p.203

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>content</td><td>[< 文字列 > | <URL> | count(< カウンタ名 >) | attr(X) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit</td><td>all</td></tr>
<tr><td>counter-increment</td><td>[< カウンタ名 > < 整数 >?]+ | none | inherit</td><td>all</td></tr>
<tr><td>counter-reset</td><td>[< カウンタ名 > < 整数 >?]+ | none | inherit</td><td>all</td></tr>
<tr><td>quotes</td><td>"[<""文字列""> <""文字列"">]+ | none | inherit"</td><td>visual</td></tr>
</table>

位置決めCSS1 ☞p.198

<table>
<tr><th>Name</th><th>Values</th><th>Media groups</th></tr>
<tr><td>clear</td><td>none | left | right | both | inherit</td><td>visual</td></tr>
<tr><td>float</td><td>left | right | none | inherit</td><td>visual</td></tr>
<tr><td>height</td><td>< 一般サイズ > | < パーセント > | auto | inherit</td><td>visual</td></tr>
<tr><td>max-width</td><td>< 一般サイズ > | < パーセント > | none | inherit</td><td>visual</td></tr>
<tr><td>min-height</td><td>< 一般サイズ > | < パーセント > | inherit</td><td>visual</td></tr>
<tr><td>min-width</td><td>< 一般サイズ > | < パーセント > | inherit</td><td>visual</td></tr>
<tr><td>width</td><td>< 一般サイズ > | < パーセント > | auto | inherit</td><td>visual</td></tr>
<tr><td>max-height</td><td>< 一般サイズ > | < パーセント > | none | inherit</td><td>visual</td></tr>
</table>

位置決めCSS2 ☞p.157

Name	Values	Media groups
bottom	< 一般サイズ > | < パーセント > | auto | inherit	visual
clip	< 形状 > | auto | inherit	visual
left	< 一般サイズ > | < パーセント > | auto | inherit	visual
overflow	visible | hidden | scroll | auto | inherit	visual
position	static | relative | absolute | fixed | inherit	visual
right	< 一般サイズ > | < パーセント > | auto | inherit	visual
top	< 一般サイズ > | < パーセント > | auto | inherit	visual
visibility	visible | hidden | collapse | inherit	visual
z-index	auto | < 整数 > | inherit	visual

ユーザインタフェース（インタラクティブ） ☞p.224

Name	Values	Media groups
cursor	[[<URL> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| text | wait | help]] | inherit	visual, interactive
outline	['outline-color' || 'outline-style' || 'outline-width'] | inherit	visual, interactive
outline-color	< 色 > | invert | inherit	visual, interactive
outline-style	<border-style> | inherit	visual, interactive
outline-width	<border-width> | inherit	visual, interactive

その他 ☞p.236

Name	Values	Media groups
direction	ltr | rtl | inherit	visual
display	inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	all
list-style	['list-style-type' || 'list-style-position' || 'list-style-image'] | inherit	visual
list-style-image	<URL> | none | inherit	visual
list-style-position	inside | outside | inherit	visual
list-style-type	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | kata	visual
marker-offset	< 一般サイズ > | auto | inherit	visual
unicode-bidi	normal | embed | bidi-override | inherit	visual
white-space	normal | pre | nowrap | inherit	visual

Index

A

B

C

D

E

F

H

L

M

N

P

S

T

U

■参考文献，URL

情報提供

W3C	http://www.w3.org/
HTML	http://www.w3.org/MarkUp/
STYLESHEET	http://www.w3.org/Style/
WAI	http://www.w3.org/WAI/
本書 support	http://www.gihyo.co.jp/css/
デザイン参考	Robin Williams 著・吉川典秀訳・米谷タツヤ解説：ノンデザイナーズ・デザインブック：（株）毎日コミュニケーションズ, ISBN4-89563-007-2 http://www.peachpit.com/peachpit/titles/catalog/48433.html

仕様書

HTML4.0	http://www.w3.org/TR/REC-html40
HTML4.0 日本語訳（内田明氏）	http://www.asahi-net.or.jp/%7Esd5a-ucd/rec-html40j/
HTML3.2	http://www.w3.org/TR/REC-html32
HTML2.0(RFC1866 と同じ)	http://www.w3.org/MarkUp/html-spec/
CSS1	http://www.w3.org/TR/REC-CSS1
CSS2	http://www.w3.org/TR/REC-CSS2
WAI-著者向け（ドラフト）	http://www.w3.org/TR/WD-WAI-PAGEAUTH
WAI-WWW ブラウザ作成者向け（ドラフト）	http://www.w3.org/TR/WD-WAI-USERAGENT/

RFC や internet-draft

原文（InterNIC）	http://www.internic.net/
ミラー（RingServer）	http://ring.aist.go.jp/pub/internet/

文法チェッカー

jweblint	http://saturn.aichi-u.ac.jp/~mimasa/jweblint/
Another HTML-lint	http://ring.aist.go.jp/openlab/k16/htmllint/index.html
CSS2 checker	http://www.w3.org/Style/CSS/Test/
CSS2 test suit	http://jigsaw.w3.org/css-validator/

■著者略歴

すみけんたろう

1974年2月24日生，愛知県出身．

小学生のときに日立の「ベーシックマスターJR.」を親戚から「おさがり」でいただいて以来の趣味のコンピュータ人間．98年3月には名古屋大学大学院農学研究科森林資源利用学研究室にて，国有林管理を対象にした長期的な森林状態把握のためのデータモデルに関する研究で修士を修めた．その裏の顔は音楽愛好家であり，フランク・ザッパ，プリンスの全アルバム感想録などをWWWで公開している“フリーク”でもある．

http://www.asahi-net.or.jp/~jy3k-sm/

カバーデザイン❖ 小倉一夫

スタイルシートWebデザイン

平成10年 8月 10日　初版 第1刷発行

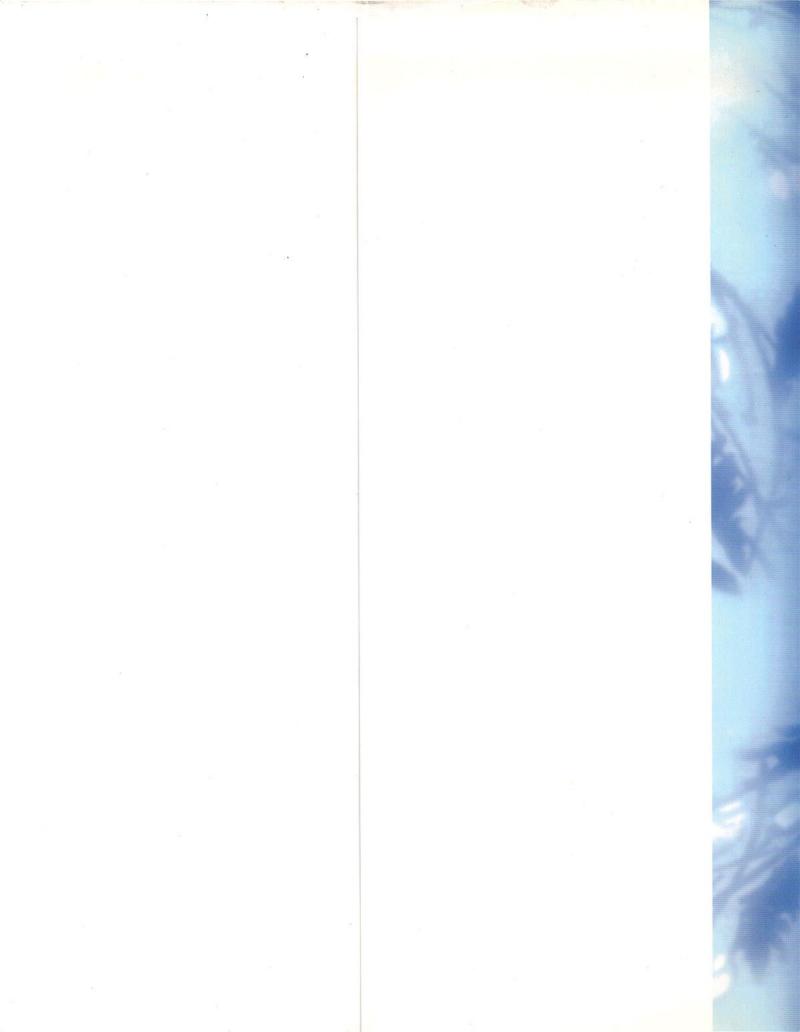
著　者　すみけんたろう
発行者　片岡　巌
発行所　株式会社技術評論社
東京都新宿区愛住町8番地8
電話　03-3225-2300　営業部
03-3225-3293　編集部
印刷／製本　株式会社加藤文明社

定価はカバーに表示してあります．

ISBN4-7741-0622-4 C3055

Printed in Japan

■ご注意

本書についての電話によるお問い合わせは、一切お受けすることはできません。質問などがございましたら、書面で弊社第2編集部までお送りくださいますよう、お願いいたします。

ISBN4-7741-0622-4 C3055 ¥1980E

定価（本体1980円十税）　M-cord 168111

Cascade Style Sheet

スタイルシート Webデザイン